Census Optimization Using Machine Learning Techniques

Fernando Jorge Vicente Marçal Liça

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Pável Pereira Calado
             Prof. Mário Jorge Costa Gaspar da Silva

Examination Committee

Chairperson: Prof. António Manuel Ferreira Rito da Silva
             Supervisor: Prof. Pável Pereira Calado
Member of the Committee: Prof. Rui Miguel Carrasqueiro Henriques

October 2018
Acknowledgments

I would like to thank my parents and grandmother for their, encouragement and support over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank sister for her support and patience to put up with me 365 days a year, given that some of them can prove quite challenging.

I would also like to acknowledge my dissertation supervisors Prof. Páve Calado and Prof. Mario Silva for their insight, support and sharing of knowledge that has made this Thesis possible.

Towards Antonio Portugal and the rest of the INE staff I would like to express my gratitude for welcoming me and providing support when possible.

A big thank you, to all my friends and colleagues that helped me grow as an individual and become a better person.

And last but not the least, a special thanks to the friend that took the time to help me review this thesis and teach me a few new facts about the English language.

To each and every one of you – Thank you.
Abstract

The objective of this dissertation is to make use of administrative data scattered between several databases and use it to improve the Portuguese Census. Using such data will reduce the time and cost necessary to perform a census possible, which may in turn allow it to happen more often and in a more reliable way. To achieve this goal a prototype was developed consisting of three components: data cleaning and normalization, indexing using standard blocking, and classification using machine learning techniques. I study several optimizations using different algorithms to increase the amount of solved conflicts and the reliability of matched pairs. The obtained results support the feasibility of the methodology and software developed for the pairing of administrative data that are now available at Statistics Portugal and shall, consequently, provide an increase in the coverage of BPR (Base da População Residente).

Keywords

Census, String Matching, Classification, Machine Learning, Blocking, Conflict Solving.
Resumo

O objectivo desta dissertação é fazer uso de dados administrativos dispersos entre várias bases de dados e utilizá-lo para melhorar o método utilizado para a realização de Census no território Português. A utilização destes dados irá reduzir o tempo e o custo necessários para a realização de census, o que, por sua vez, pode permitir que isto aconteça com uma frequência e de forma mais confiável. Para atingir este objetivo, foi desenvolvido um protótipo composto de três componentes: limpeza e normalização de dados, indexação usando standard blocking e classificação usando técnicas de aprendizagem automática. Eu testo várias otimizações usando algoritmos diferentes para aumentar a quantidade de conflitos resolvidos e a confiabilidade dos pares emparelhados. Os resultados obtidos suportam a viabilidade desta metodologia e do software desenvolvido para o emparelhamento de dados administrativos que estão agora ao dispor do INE o que, consequentemente, irá aumentar a cobertura da BPR (Base da População Residente).

Palavras Chave

Census, Emparelhamento de Strings, Aprendizagem Automática, Classificação, Blocagem, Resolução de Conflitos;
# Contents

1 Introduction

1.1 Introduction ........................................... 3
1.2 Background ........................................... 3
1.3 Objective ............................................. 4
1.4 Methodology .......................................... 5
1.5 Contributions ......................................... 6
1.6 Document Organization ................................. 7

2 Basic Concepts

2.1 Basic Concepts ......................................... 11
2.2 Record Linkage ........................................ 11
2.3 Blocking ............................................... 12
2.4 String Similarity ....................................... 14
  2.4.1 Levenshtein Distance ............................ 14
  2.4.2 Jaccard ............................................. 15
  2.4.3 Jaro-Winkler ....................................... 16
  2.4.4 Hamming Distance ............................... 18
  2.4.5 Sørensen-Dice Coefficient ..................... 18
2.5 Classification .......................................... 19
  2.5.1 Logistic Regression ............................... 20
  2.5.2 Support Vector Machines ....................... 20
  2.5.3 Decision Trees .................................... 21
  2.5.4 Naive Bayes ....................................... 22
  2.5.5 Random Forest .................................... 23
2.6 Conflict Solving ....................................... 23

3 Related Work

3.1 Related Work ........................................... 31
3.2 Duplicate Detection ................................. 31
# List of Figures

1.1 European Countries Census Type in 2010 ............................................. 4
2.1 Schema of the Record Linkage Process................................................. 12
2.2 Example of a Decision Tree ................................................................. 22
2.3 Example of a Random Forest ................................................................. 24
2.4 Conflict Resolution Categories .............................................................. 26
3.1 UML diagram of entities and relationships ............................................. 33
3.2 Beyond 2011 Methodology .................................................................... 37
4.1 Record Linkage System Architecture ..................................................... 41
List of Tables

1.1 United States population census costs in the last century. ............................... 3

2.1 Example of records. ......................................................................................... 13

2.2 Blocking Keys ................................................................................................. 13

2.3 Example of records with the same blocking key. .............................................. 13

2.4 Match Pair. ...................................................................................................... 19

2.5 Non-Match Pair. ............................................................................................... 19

2.6 Example of data fusion conflicts. ..................................................................... 25

3.1 Example of BPR with personal identifiers/keys of different databases. .......... 34

4.1 Possible Matches not yet present in BPR. .......................................................... 42

5.1 This table shows which fields are present in each database. ............................ 46

5.2 Classification possible outcomes. ..................................................................... 46

6.1 Differences between the amount of new matches found. ................................ 51

6.2 Differences between the new matches found with 260 data and 2015 data. ....... 52

6.3 Accuracy for the string matching algorithms .................................................. 53

6.4 Accuracy for the Classification Algorithms ..................................................... 54

6.5 Comparison between the results from previous solution and new solution. ........ 54

6.6 Number of Civil Population Register (BDIC) Records from 2016. ................. 56

6.7 Analysis on Completeness, Number of Null Records and Distant Values. ........ 56

6.8 Reduction Ratio using blocking for the pair BDIC-General Retirement Fund (CGA). 56

6.9 Quality analysis of the classification model. .................................................... 57

6.10 Number of new matches obtain by pairing BDIC with CGA from 2016. .......... 57

6.11 Analysis on the Contradictions and Uncertainties. ....................................... 57
# Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIC</td>
<td>Civil Identification Number</td>
</tr>
<tr>
<td>NIF</td>
<td>Fiscal Identification Number</td>
</tr>
<tr>
<td>NISS</td>
<td>Social Security Number</td>
</tr>
<tr>
<td>INE</td>
<td>Statistics Portugal</td>
</tr>
<tr>
<td>BDIC</td>
<td>Civil Population Register</td>
</tr>
<tr>
<td>AT</td>
<td>Tax Authority</td>
</tr>
<tr>
<td>IISS</td>
<td>Informatics of Social Security Institute</td>
</tr>
<tr>
<td>EDUC</td>
<td>General Statistics of Education and Science</td>
</tr>
<tr>
<td>CGA</td>
<td>General Retirement Fund</td>
</tr>
<tr>
<td>IEFP</td>
<td>Unemployment and Vocational Training Institute</td>
</tr>
<tr>
<td>SEF</td>
<td>Immigration and Borders Service</td>
</tr>
<tr>
<td>BPR</td>
<td>Resident Population Base</td>
</tr>
<tr>
<td>BKV</td>
<td>Blocking Key Values</td>
</tr>
<tr>
<td>SCA</td>
<td>Sequential Covering Algorithm</td>
</tr>
<tr>
<td>RR</td>
<td>Reduction Ratio</td>
</tr>
<tr>
<td>PC</td>
<td>Pair Completeness</td>
</tr>
<tr>
<td>UKSA</td>
<td>UK Statistics Authority</td>
</tr>
<tr>
<td>HESA</td>
<td>Higher Education Statistics Agency</td>
</tr>
<tr>
<td>NHS</td>
<td>National Health Service</td>
</tr>
</tbody>
</table>
Introduction

Contents

1.1 Introduction ......................................................... 3
1.2 Background ......................................................... 3
1.3 Objective .......................................................... 4
1.4 Methodology ......................................................... 5
1.5 Contributions ......................................................... 6
1.6 Document Organization ............................................. 7
1.1 Introduction

A National census is one of the most important sources for statistical and socio-economic information. A census provides a basis for the official statistics of a country, including the statistical information about its population, ranging from degree of literacy to the number of families. This work expands and optimizes a previous solution that uses machine learning techniques to handle this information in an acceptable computational time window.

1.2 Background

A census is a complex task that requires meticulous planning of every aspect, both methodological and technological, which will without a doubt mean high expenses. As an example, let’s take a look at Table 1.1, where the costs of the population census in the United States in the last century are represented Gauthier [2002]:

<table>
<thead>
<tr>
<th>Census Year</th>
<th>Total Population</th>
<th>Census Cost</th>
<th>Average Cost Per Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>1910</td>
<td>91,972,266</td>
<td>$15,968,000</td>
<td>17.07 cents</td>
</tr>
<tr>
<td>1920</td>
<td>105,710,620</td>
<td>$25,117,000</td>
<td>23.76 cents</td>
</tr>
<tr>
<td>1930</td>
<td>122,775,046</td>
<td>$40,156,000</td>
<td>32.71 cents</td>
</tr>
<tr>
<td>1940</td>
<td>131,669,275</td>
<td>$67,527,000</td>
<td>51.29 cents</td>
</tr>
<tr>
<td>1950</td>
<td>151,325,798</td>
<td>$91,462,000</td>
<td>60.44 cents</td>
</tr>
<tr>
<td>1960</td>
<td>179,323,175</td>
<td>$127,934,000</td>
<td>71.34 cents</td>
</tr>
<tr>
<td>1970</td>
<td>203,302,031</td>
<td>$247,653,000</td>
<td>1.22 dollars</td>
</tr>
<tr>
<td>1980</td>
<td>226,542,199</td>
<td>$1,078,488,000</td>
<td>4.76 dollars</td>
</tr>
<tr>
<td>1990</td>
<td>248,718,301</td>
<td>$2,492,830,000</td>
<td>10.02 dollars</td>
</tr>
<tr>
<td>2000</td>
<td>281,421,906</td>
<td>$4.5 Billion</td>
<td>15.99 dollars</td>
</tr>
<tr>
<td>2010</td>
<td>308,745,538</td>
<td>$13 Billion</td>
<td>42.11 dollars</td>
</tr>
</tbody>
</table>

Table 1.1: United States Population Census Costs In The Last Century.¹

As we can see, these costs grow along side the population and can amount to considerably high sums of money. It’s worth noting that the average cost per person is not adjusted to the inflation, in reality they should be lower, but this still gives us an idea about the cost growth, which brings us to one of the main reasons to why population census don’t happen more frequently. Due to that fact, several European countries are committed to migrate from the traditional census model to a new one based on citizen’s administrative registers Baffour et al. [2013], Valente [2010].

¹http://www.genealogybranches.com/censuscosts.html
The idea behind administrative based census is that every modernized country should have at least one digital database where its citizens are registered, so instead of sending an agent to every home in order to count how many people live there, like in the traditional model, the register-base model looks instead in these databases for data about their citizens, such as their age and current living address, for example. Ultimately the population characteristics considered are limited to those available in the registers, therefore it's necessary to combine data across different databases in order for it to become high quality data Valente [2010].

This new type of census is starting to gain followers especially amongst the northern countries. As we can see in Figure 1.1, in 2010, countries like Norway, Sweden, Finland and Denmark already used register based census, while countries like France, Germany, Poland and Czech Republic use an hybrid combination between registers and more traditional means like forms and interviews.

1.3 Objective

The objective of this dissertation is to improve upon the work of Silva [2017] and Velho [2017] by testing new algorithms and optimizations in a system previously developed, to allow a census based on administrative data provided by the following entities:

\[\text{www.tilastokeskus.fi/tup/v12010/art_2011-05-17_001_en.html}\]
These optimizations consist in testing several new algorithms for the main phases of the process, these being: different string matching algorithms, while measuring their performance and new machine learning algorithms, to improve the classification phase. I also expand the previous work by testing new ways to match records, for example testing new blocking key. This improves INE’s record quality by reducing the amount of inconsistencies and errors present in Resident Population Base (BPR) enabling them to use these records for an administrative based census. This, in turn, will enable Portugal to perform population census more often and have a positive socio-economical impact nation wide.

1.4 Methodology

The applied methodology started by trying to understand the problem and the work environment with the INE staff at its headquarters in Alameda. I signed a confidential agreement due to the access of sensitive data which meant I could not work remotely and was limited to work directly at INE. I was provided a special laptop with all the tools used in the previous project. The pre-installed tools where: Oracle, Python language with libraries such as Numpy and Scikit Learn, X-Ming to give me a graphical interface to the server, which allow me to navigate the project folders like I would in the windows explorer and Putty to make the connection between the server, where all the databases where stored, and my
local work space. I began by accessing the files to learn how the previous project was organized and performing a few queries to create sample tables with approximately a hundred records, to check if I had the necessary privileges. While everything was set up, I started planning what tools and algorithms I should test, and decided that I would experiment with Levenshtein, Jaccard, Jaro-Winkler, Hamming and Dice for string matching and Logistic Regression, Support Vector Machines, Decision Trees, Naive Bayes and Random Forest to classify the match candidates. Like it was previously discussed with INE, I would begin by performing tests with the 2015 data in order to replicate the results achieved by Silva [2017] and Velho [2017], and only after finishing these tests would I then move to the 2016 data. During this period, I also concluded that I should re-test the generated model with a different amount of folds to determine if the current 2-fold validation is a good choice. Several alternatives to the blocking key were also tested, yet the results were negative in both cases, increasing the time required to run the process (the time difference was not measured because these changes meant a jump from a few minutes to at least 3 hours in execution time, at which point I terminated the experience). Afterwards I calculated the accuracy using different combinations of the algorithms that I previously mentioned, and the pair Jaccard - Decision Trees managed to score the highest accuracy percentage.

### 1.5 Contributions

This work’s results are: the implementation of four new string matching algorithms (Jaccard, Jaro-Winkler, Dice and Hamming) and four new classification algorithms (Decision Trees, Support Vector Machines, Random Forest and Naive Bayes) that allow a more flexible and complete testing environment for record matching, fixes to many of the remaining problems from the previous project, like fixing the ability to generate the completeness and uncertainty reports, while expanding its scope, by introducing new features like a Python script that can generate SQL queries for table creation without the necessity of configuration the files previously used. I also investigated if the changes to the blocking keys and number of folds (for cross validation) chosen in the previous project bring any significant improvements and reach the conclusion that the current values (detailed in further sections) are the optimal choice, and came to the conclusion that they are indeed the best choice. I then performed a series of tests to find the combination of algorithms that can achieve an higher level of accuracy compared to the ones used in the previous work. The test showed me that the two algorithms that performed better were Jaccard and Decision Trees: Jaccard distance (using Log. Regression for the classification) achieved a higher accuracy in 75% of the database pairs and the Decision Trees classifier (using Jaccard for string matching) achieved a higher accuracy in 50% of the database pairs. With this thesis, I ultimately improve upon the project started by Silva [2017] and Velho [2017] by expanding its functionality and making it
more reliable. This will hopefully prove to be useful to INE as the 2021 census gets closer, allowing it to be one step closer to the future of register based census.

1.6 Document Organization

This document is organized in the following way:

1. Section 2, focuses on the basic concepts necessary to properly understand how we will achieve our results, how to replicate them and what they mean.

2. Section 3, makes reference to previous works, making a brief description of what they accomplished and why are they important to this thesis.

3. Section 4, makes a description of the proposed work.

4. Section 5, makes a summary of the accomplished tasks, explains the origins of the data and which processes I use to evaluate my results.

5. Section 6, give a short conclusion for this work.

6. Section 7, provides some thoughts about the improvements for future work.
Contents

2.1 Basic Concepts .................................................. 11
2.2 Record Linkage .................................................. 11
2.3 Blocking ......................................................... 12
2.4 String Similarity ................................................ 14
2.5 Classification ..................................................... 19
2.6 Conflict Solving ............................................... 23

Basic Concepts
2.1 Basic Concepts

In order to have a better understanding of the process I use to tackle this complex problem, let’s first take a look at the algorithms that lie at its core. The previous system is composed of a combination of the most well-known algorithms in record linkage I. P. Fellegi [1969], and the challenge is to understand which is the combination of these methodologies that will achieve the best results, in the most efficient way. But before we get to that, it is important to have a comprehensive understanding of all the basic concepts, like Duplicate Records, Blocking, String Similarity, Classification and Data Evaluation Metrics, Conflict Solving and Record Linkage itself, that serves as the basis for this dissertation.

2.2 Record Linkage

Record Linkage, as the name indicates, is the task of finding two records in two different databases that represent the same real world entity. It may be the case that the records in each database are different, yet they represent the same entity or, conversely, they may be equal but actually represent different entities. This can happen due to errors or due to the fact that there aren’t sufficient characteristics included in the records I. P. Fellegi [1969].

When two different representations (records) of a real world entity exist due to these records containing errors, these different records are called Duplicates. When joining data provided by several different sources, duplicates are bound to happen. Cleaning and normalizing data might help to avoid this problem to a certain extent, but some errors will persist, hence the necessity of having a matching generator to do this automatically.

In Figure 2.1, we can see the scheme of the record linkage process. It begins by taking the data from information sources, in this case Database A and Database B, which goes through the Cleaning and Standardization phase where the raw data is modified, replaced or even deleted if it’s inconsistent or incorrect, turning it into reliable data. In addition, the data will also be normalized, to ensure that it has the same consistent format across all databases.

Next we have the Indexing phase, where the records that are going to be paired and compared are selected. This step is necessary due to the high quantity of records. Let’s say we have Relation A and Relation B each with 1,000 records. To obtain the matching candidates of these two relations we would need to perform $1,000 \times 1,000 = 1,000,000$ comparisons. If we consider the fact that the task I’m trying to accomplish has millions of records, it would be computationally impracticable to perform every single comparison. To speed up this process, Blocking techniques are used to reduce the number of comparisons required. This will be further explained in Section 2.3.
In the **Record Pair Comparison** phase, we take the records previously selected in the indexing phase and compare them using string matching techniques further explained in Section 2.4.

The **Similarity Vector Classification** phase classifies the records as Matches, Non-matches or Possible matches using similarity scores, and in the **Clerical Review** phase the possible matches can be labeled as Matches or Non-Matches by an expert.

Finally, in the **Evaluation** phase, the retrieved results will go through an evaluation so that the parameters can be further adjusted to obtain the best possible result.

### 2.3 Blocking

The **Traditional Standard Blocking** technique is a widely used way to speed up record matching since the 60’s I. P. Fellegi [1969]. It consists in generating a blocking key for each record, grouping together similar records in blocks and comparing only the records that are inside these blocks. Let’s us take a look at an example in Table 2.1.

As we can see there are a few inconsistencies. The first in the address field where St. Antônio is a abbreviation of Santo António and the second in the last name field where there are two different values,
possibly due to an error while inserting the record in the database.

Let’s now assume our blocking criteria, which essentially is the rule that allows the algorithm to determine how the blocking key is formed, is a combination of the last name and birth date fields. This will originate the following blocking keys represented in Table 2.2.

For the first two records, the difference is in the address field. Due to the fact that this particular field is not part of our blocking criteria both records will end up having the same blocking key. The same cannot be said for the two last two records, since the last name field is part of our blocking criteria therefore it will create two different blocking keys for each one, as we can see in Table 2.3.

The Sorted Neighbor Blocking technique starts by sorting the records by their blocking key values Christen [2012b]. A window of a fixed size $w > 1$ is then moved over the sorted records and the candidate record pairs are generated from the records inside the window. This technique can be implemented in two different ways:

Through the Sorted Array-Based Approach, that inserts the blocking key values into an array of size $n_A + n_B$ with $n_A$ and $n_B$ representing the number of records in Database A and Database B. The window is then moved over this sorted array generating candidate record pairs from all records in the current window that should have $(n_A + n_B + w + 1)$ positions, the number of candidate record pairs generated in the window’s first position is $\frac{n_A+n_B}{(n_A+n_B)^2} \cdot w^2$ while the number of candidate record pair generated by the remaining positions is $\frac{n_A+n_B}{(n_A+n_B)^2} \cdot 2(n_A + n_B - w)(w - 1)$. The total number of unique candidate record pairs is given by the following Equation 2.1

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
<th>Birth Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedro</td>
<td>Gonçalves</td>
<td>Santo António</td>
<td>02-04-78</td>
</tr>
<tr>
<td>Pedro</td>
<td>Gonçalves</td>
<td>St. António</td>
<td>02-04-78</td>
</tr>
<tr>
<td>José</td>
<td>Oliveira</td>
<td>Nazaré</td>
<td>18-06-84</td>
</tr>
<tr>
<td>José</td>
<td>Olivais</td>
<td>Nazaré</td>
<td>18-06-84</td>
</tr>
</tbody>
</table>

**Table 2.1:** Example of records.

<table>
<thead>
<tr>
<th>Blocking Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gonçalves020478</td>
</tr>
<tr>
<td>Gonçalves020478</td>
</tr>
<tr>
<td>Oliveira180684</td>
</tr>
<tr>
<td>Olivais180684</td>
</tr>
</tbody>
</table>

**Table 2.2:** Blocking Keys for the records of Table 2.

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
<th>Birth Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedro</td>
<td>Gonçalves</td>
<td>Santo António</td>
<td>02-04-78</td>
</tr>
<tr>
<td>Pedro</td>
<td>Gonçalves</td>
<td>St. António</td>
<td>02-04-78</td>
</tr>
</tbody>
</table>

**Table 2.3:** Example of records with the same blocking key.
\[ CP = \frac{n_A \times n_B}{(n_A + n_B)^2} \left( w^2 + 2(n_A + n_B - w)(w - 1) \right) \]  

(2.1)

It's important to notice that, due to the fact that the windows size is fixed in this approach, the number of candidate record pairs generated is independent of the frequency distribution of the Blocking Key Values (BKV), and only depends upon the window size \( w \) and the size(s) of the database(s) Christen [2012b].

Sorted neighbor blocking can also be implemented through the Inverted Index-Based Approach that, instead of inserting the BKV in a sorted array, it stores the blocking key values in an inverted index\(^1\). The window then moves over these values and candidate record pairs are formed from all records in the corresponding index lists. The number of windows positions is given by \((b - w + 1)\) where \( b \) is the number of different blocking key values. Considering these have a uniform distribution, each inverted index list will contain \( \frac{n_A}{b} + \frac{n_B}{b} \) record identifiers. The number of candidate record pairs is given by the following Equation:

\[ CP = \frac{n_A \times n_B}{b^2} \left( w^2 + (b - w)(2w - 1) \right) \]  

(2.2)

\[ \]  

2.4 String Similarity

Records are composed by several fields like name, age, sex, address etc, so in order to compare two of them we need to compare the strings in both records for each field. The String Similarity metric, as its name suggests, measures the similarity between two strings. The closest the strings are to each other, the higher their similarity is.

To compare these strings, we’ll use String Similarity algorithms. There are several algorithms to calculate this similarity, but we’re only going to talk about a few of the most well known amongst them.

2.4.1 Levenshtein Distance

Also known as Edit Distance, it’s essentially the number of operations (insertions, deletions and substitutions) needed to transform one string into another. Each operation has the cost of one, according to Levenshtein [1966]. To calculate the Edit Distance we can build a matrix \( d(i,j) \) in the following way:

\[ \]  

\[ \]
\[d(i,j) = \begin{cases} 
    d(i-1,j-1) + c(x_i, y_i) & \text{copy or substitute} \\
    d(i-1,j) & \text{delete } x_i \\
    d(i,j-1) & \text{delete } y_i
\end{cases}\] (2.3)

Where:
\[c(x_i, y_i) = \begin{cases} 
    0 & \text{if } x_i = y_i, \ 1 \text{ otherwise}
\end{cases}\]
\[d(0, 0) = 0,\]
\[d(i, 0) = i,\]
\[d(0, j) = j.\]

To more easily demonstrate how the similarity score is calculated we can use these two strings: Sun and Sunny.

When we apply the algorithm we end up with a distance of 2, for this particular example the only differences between the two strings are the two extra characters present in "sunny", "n" and "y" respectively.

### 2.4.2 Jaccard

The Jaccard similarity is one of the simplest algorithms we can apply to the problem. The idea behind it is the following:

Let’s take a string \(x = \text{ sunday} \) and a string \(y = \text{ monday}\)

First we split each string in q-grams, which are a contiguous sequence of q characters of a string. For example, a bigram (q=2) creates a window of two characters that will slide over the string, while a trigram (q=3) creates a window of three characters.

For this algorithm we will start by dividing the string \(x\) and \(y\) in bigrams to obtain the following results:

\[x = [su],[un],[nd],[da],[ay]\]
\[y = [mo],[on],[nd],[da],[ay]\]

The Jaccard similarity is defined by the equation 2.4:
\[ Jaccard(x, y) = \frac{|x \cup y|}{|x \cap y|} \]  

(2.4)

In equation 2.4, \(|x \cup y|\) represents the number of common q-grams between \(x\) and \(y\), while \(|x \cap y|\) represents the size of the union between the q-grams of \(x\) and \(y\).

So for this specific example, between string \(x\) and \(y\) we would obtain a similarity of:

\[ Jaccard(x, y) = \frac{3}{7} = 0.428 \]

2.4.3 Jaro-Winkler

**Jaro-Winkler** is a string distance measurement intended primarily for short strings Cohen et al. [2003]. In order to understand this algorithm we first need to take a look at **Jaro-Distance** which serves as the base where **Jaro-Winkler** is built upon:

\[ Jaro(x, y) = \frac{1}{3} \left( \frac{c}{|x|} + \frac{c}{|y|} + \frac{c - t}{c} \right) \]  

(2.5)

Where:

- \(|x|\) is the length of string \(x\).
- \(|y|\) is the length of string \(y\).
- \(c\) is the number of common characters between the two strings.
- \(t\) is the number of transpositions between common characters.

Two characters are considered common if:

\[ x_i = y_j \]

and

\[ |i - j| \leq \frac{\min(|x|, |y|)}{2} \]  

(2.6)

Where:

- \(x_i\) is the character in position \(i\) of the \(x\) string.
\( y_j \) is the character in position \( j \) of the \( y \) string.

For example, let’s look at this two strings:

\[
x = \text{TRACE} \\
y = \text{CRATE}
\]

The common characters between \( x \) and \( y \) are \([R,A,E]\) therefore \( m = 3 \). The characters \([T,C]\) aren’t considered common characters because even though they exist in both strings they do not respect the second condition in Equation 2.6:

\[
|1 - 4| \leq \frac{\min(5,5)}{2} \equiv 3 \leq 2.5
\]

Finally moving on to Jaro-Winkler, we only need to introduce two new parameters: \( PL \) (Prefix Length) is the length of the longest common prefix between the two strings, and \( PW \) (Prefix Weight), which stands for the weight of said prefix. The standard value for the prefix weight is 0.1 and it should never go above 0.25, otherwise the distance may become larger than 1 Gurneet Kaur [2014]. This modification is based on the fact that there are more errors in the last names Yancey [2005]. This algorithm is represented by the following Equation 2.7:

\[
Jaro - Winkler(x, y) = (1 - PL \times PW) \times Jaro(x, y) + PL \times PW
\]

Let’s look at a new example:

\[
x = \text{MARTHA} \\
y = \text{MARHTA}
\]

In this case we would have:

\[
Jaro(martha, marhta) = \frac{1}{3} \left( \frac{6}{6} + \frac{6}{6} + \frac{6 - 1}{6} \right) = 0.944
\]

Three initial characters match, MAR, for a Jaro-Winkler distance of:

\[
Jaro - Winkler(martha, marhta) = (1 - 3 \times 0.1) \times 0.944 + 3 \times 0.1 = 0.961
\]
2.4.4 Hamming Distance

The *Hamming Distance* is one of the simplest string matching algorithms: when comparing two strings, it will count the minimum number of errors that could have transformed one string into the other. Let us consider the following two strings:

\[ x = \text{BERNIE2020} \]
\[ y = \text{BARNES2020} \]

The *Hamming Distance* for this two strings is 3 as there are three characters that are different between the two.

2.4.5 Sørensen-Dice Coefficient

*Sørensen-Dice’s* coefficient measures how similar a set and another set are. It can be used to measure how similar two strings are in terms of the number of common bigrams (a bigram is a pair of adjacent letters in the string). This coefficient is represented by the following Equation 2.10:

\[
S_\text{sørensen-Dice}(x, y) = \frac{2N_t}{N_x + N_y}
\]  

(2.10)

Where \( N_t \) is the number of character bigrams found in both strings, \( N_x \) is the number of bigrams in string \( x \) and \( N_y \) is the number of bigrams in string \( y \). For example, to calculate the similarity between:

\[ x = \text{BRIGHT} \]
\[ y = \text{BRACHT} \]

The bigrams that result from each word are:

\[ x = (br, ri, ig, gh, ht) \]
\[ y = (br, ra, ac, ch, ht) \]

Each bigram set has five elements, and the intersection of these two sets has two elements: \( br \) and \( ht \). Inserting these numbers into the formula, we calculate:
Sørensen-Dice(x,y) = \frac{2 \times 2}{5 + 5} = 0.4.

2.5 Classification

I will make use of Supervised Learning techniques from section 2.5.1 to classify if two records are either a Match or a Non-Match. Thus I will make a brief, conceptual explanation of what is Supervised Learning and what techniques we can use.

<table>
<thead>
<tr>
<th>Database</th>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Miguel</td>
<td>Marques</td>
<td>Póvoa de Santa Iria</td>
</tr>
<tr>
<td>B</td>
<td>Miguel</td>
<td>Marques</td>
<td>Póvoa de Santa Iria</td>
</tr>
</tbody>
</table>

Table 2.4: Match Pair.

<table>
<thead>
<tr>
<th>Database</th>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Miguel</td>
<td>Marques</td>
<td>Póvoa de Santa Iria</td>
</tr>
<tr>
<td>B</td>
<td>Pedro</td>
<td>Melo</td>
<td>Santarém</td>
</tr>
</tbody>
</table>

Table 2.5: Non-Match Pair.

Let’s start with something simple: we have two classes, Match and Non-Match, and our features shall be First Name, Last Name and Address, as shown in Tables 2.4 and 2.5. We are then given two sets of record pairs: in the first one the pairs are already labeled, so we already know their classes, either a Match or Non-Match. This is going to be our Training Set. In the second set, the pairs are not labeled and we have to find a way to classify them, this will serve as the Test Set. This is a classic supervised learning problem, given we already have a set of classified record pairs as our training set, we can now submit it to a supervised algorithm to obtain a model. We can then apply that model to our unlabeled record pair set (Test Set) in order to classify it.

There are several possible supervised learning algorithm but we are going to be focused on Logistic Regression Peng et al. [2002], Support Vector Machines (SVM) Burges [1998], Decision TreesQuinlan [1986], Naive BayesKevin Drebler [2015] and Random ForestBreiman [2001], of which I will give a more detailed explanation.
2.5.1 Logistic Regression

Logistic Regression is a Supervised Learning method for analyzing a dataset in which there are one or more independent features that will determine the outcome Peng et al. [2002]. In this algorithm the outcome is binary, only taking values between 0 and 1.

To do this, the Logistic Regression method utilizes the logistic function, also known as, sigmoid function, that can take any real input \( t \), \( t \in \mathbb{R} \) and return a value that is always between 0 and 1, allowing it to be interpreted as a probability:

\[
\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{2.11}
\]

We can consider \( t \) as a linear function which usually takes the form \( t = mx + b \). We can then re-write this as \( t = \beta_0 + \beta_1 x \), which in turn let us re-write the logistic function as it follows:

\[
F(x) = \frac{1}{1 + e^{-\beta_0 + \beta_1 x}} \tag{2.12}
\]

In essence, \( \beta_0 \) and \( \beta_1 \) will act as weights that can be estimated by calculating the maximum-likelihood of the training set.

So to put this into context let us say we were trying to predict if a pair of records is in fact a match, and for simplicity’s sake let us assume \( \beta_0 = 3 \) and \( \beta_1 = 4.5 \). The probability of two pairs being a match given the similarity between their last name is 0.8 is:

\[
F(x) = \frac{1}{1 + e^{3+4.5\times0.8}} = 0.998 \tag{2.13}
\]

Which means that this two records are, most likely a match (in reality we need to consider more that just the last name to classify a pair of records as a match).

2.5.2 Support Vector Machines

The SVM algorithm can be used in both classification and regression problems, although it’s mainly used for classification problems Burges [1998]. This algorithm takes a training set with data that can be in a \( n \)-dimensional space \((n\) being the number of features) and tries to classify each point by finding the
hyper-plane that better differentiates the two classes. The optimal hyperplane is the one that maximizes the margin between the nearest training-data point of any class (called functional-margin). Let’s now formally define the hyperplane with Equation 2.14:

\[ f(x) = \beta_0 + \beta^T x \]  

(2.14)

where \( \beta \) is the weight vector, \( \beta_0 \) is the bias and \( x \) is the representation of the training examples that are closest to the hyperplane, called support vectors.

We want to maximize the margin which is equivalent to minimize the function \( L(\beta) \).

\[
\min_{\beta, \beta_0} L(\beta) = \frac{1}{2} \|\beta\|^2 \quad \text{subject to } y_i(\beta_0 + \beta^T x) \geq 1 \forall i 
\]

(2.15)

The above is an optimization problem with a convex quadratic objective and only linear constraints. Its solution gives us the optimal margin classifier. This optimization problem can be solved using quadratic programming (QP) or with the lagrange duality that will lead us to our optimization problem’s dual form, allowing us to derive an efficient algorithm for solving the above optimization problem that will typically do much better than generic QP\(^2\).

2.5.3 Decision Trees

Decision Trees are a supervised learning method used for classification and regression that can help to support decisions and predict outcomes. The model is simple to understand by humans and performs well with large data sets, while remaining fast. In a decision tree structure there are three parts: non-leaf nodes, branches and leaf nodes (terminals nodes).

The root (the topmost node) and the others non-leaf nodes work as a test on an feature. From these nodes there are branches that represent the results of the test, which could follow to another non-leaf node or a terminal node. The terminal nodes are the class labels. To clarify, given a datapoint \( X \), its feature values are used on the decision tree and a path is traced until reaching a class prediction.

The order of the features used along the way its decided by how much information we gain from each feature, the information gain can be obtained by the formula 2.16:

\[
\text{Gain}(T, X) = \text{Entropy}(T) + \text{Entropy}(T, X)
\]

(2.16)

\(^2\)www.cs229.stanford.edu/notes/cs229-notes3.pdf  
^3www.chunml.github.io/ChunML.github.io/tutorial/Decision-Tree/
with $T$ representing our class and $X$ representing a feature. For the example present in Figure 2.2, $T$ would be Yes or No and $X$ could be Weather or Temperature. Further explanation can be found in the work of Mitchell [1997] and here in this website.

2.5.4 Naive Bayes

It is a classification technique based on Bayes’s Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. As in the following Equation 2.17:

$$P(c \mid x) = \frac{P(x \mid c) * P(c)}{P(x)}$$

(2.17)
Where:

\[ P(c|x) \] is the posterior probability of class c given predictor x.
\[ P(c) \] is the prior probability of class.
\[ P(x|c) \] is the likelihood which is the probability of predictor given class.
\[ P(x) \] is the prior probability of predictor.

For this project I used the Gaussian Naive Bayes classifier given the fact that the Bernoulli Naive Bayes classifier is used when we have binary features, and the features I had to work with were not binary as they can take every value between 0 and 1. The Multinomial Naive Bayes is more suited for frequency related features which led me to conclude that the Gaussian classifier was the best choice.

### 2.5.5 Random Forest

Random Forest is a collection of Decision Trees, but there are some differences. If we input a training dataset with features and labels into a decision tree, it will formulate some set of rules, which will be used to make the predictions Breiman [2001].

For example, if we want to predict whether a pair of records are a match or not, we could select the records that we considered to be matches and collect the features that helped us to make that decision in the past (for example the to records having the same name and the same birth date). If we put the features and labels into a decision tree, it will generate some rules. Then you can predict whether the two records are a match or not. In comparison, the Random Forest algorithm randomly selects observations and features to build several decision trees and then averages the results.

Another difference is that “deep” decision trees might suffer from overfitting. Random Forest prevents overfitting most of the time, by creating random subsets of the features and building smaller trees using these subsets. Afterwards, it combines the subtrees. Note that this doesn’t work every time and that it also makes the computation slower, depending on how many trees your random forest builds.

### 2.6 Conflict Solving

By taking data from different sources and taking into account the size of the databases, conflicts between records are expected to happen. Therefore a solution is needed to guarantee that a record
in one database corresponds to a unique record in all the remaining ones. When data is taken from several heterogeneous sources and we try to fuse them together two types of inconsistencies are bound to happen in the process according to Bleiholder and Naumann [2006]:

1. **Schematic Inconsistencies**: happen when a registry has different attributes, when the data is stored in different structures or when the registry is semantically the same but with different names.

2. **Data Inconsistencies**: occur when there are several values used to describe the same property of an entity in the real world.

It is important to keep in mind the fact that INE has several requirements for their variables due to privacy concerns such as:

1. Each individual name is represented by the first 3 letter of its first name and the last 3 letter of its last name.

2. Numerical identifiers like (NIF) ans (NISS) and (AR), are cyphered with a SHA256 hash.

3. Birth date is converted to a single numerical value for example 03/10/1993 becomes 03101993.
4. Sex is identified by numerical values 1 for Males and 2 for Females.

5. Each data source as an additional attribute called “ANO” (year) that as its name indicates, it identifies the year the data belongs to.

6. Postal code is divided in 2 attributes, one for its first 4 digits and the other for the last 3 digits.

These requirements raise other issues that need to be taken into account such as:

- Not every registry in the databases have a common personal identifier.
- Individuals might not even be registered in every database.
- There are errors and inconsistencies since most of the data was manually introduced.

Therefore data inconsistencies techniques are used to solve them. There are two types of data inconsistencies: **Contradictions** and **Uncertainties**. To better understand them let's take a look at Table 2.6:

<table>
<thead>
<tr>
<th>Relation</th>
<th>Id</th>
<th>Name</th>
<th>Sex</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>77207</td>
<td>José</td>
<td>M</td>
<td>Oeiras</td>
</tr>
<tr>
<td>B</td>
<td>77270</td>
<td>José</td>
<td>Null</td>
<td>Oeiras</td>
</tr>
<tr>
<td>Fusion</td>
<td>?</td>
<td>José</td>
<td>?</td>
<td>Oeiras</td>
</tr>
</tbody>
</table>

**Table 2.6:** Example of data fusion conflicts.

**Contradictions** happen when there is a conflict between two non-null values. As we can see in Table 2.6 the Id’s of record A and record B are non-null but are different from each other which may have been caused by a human mistake when introducing the values in the database.

**Uncertainties** originate from a conflict between a non-null value and one or more null values. In Table 2.6 record A Sex attribute is conflicting with registry’s B Sex attribute which is null. When there is a null value in a database, it may happen for one of three reasons: whoever entered the data did not know the value, the data exists but we don’t have the permissions to see it or we cannot apply the corresponding property to this specific object Bleiholder and Naumann [2006].

According to Bleiholder and Naumann [2006] conflict resolution can be divided in three main categories as shown in Figure 2.4.

1. **Conflict Avoidance**: handles data inconsistencies even though it does not solve the individual conflicts. This category has three strategies that can either be Instance Based or Meta-data Based.
(a) **Take the Information** is an instance based strategy. When facing a uncertainty between a null and a non-null value it will always take the non-null value. This strategy only works when we are facing a uncertainty.

(b) **No Gossiping** is a instance based strategy that only takes into account consistent answers that fulfill a constraint on the query, leaving aside all inconsistent ones.

(c) **Trust Your Friends** is a meta-data based strategy where we choose in whom we trust. In other words, we choose the source that we consider more reliable. Once we make this decision it will be carried out for all values not taking into account if there is a inconsistency or not.

2. **Conflict Ignorance**: describes strategies that are simple to implement and don’t even require awareness of the data conflicts in the data because they don’t take any decision at all.

   (a) **Pass It On** takes all the conflicting values and passes them to either a human or a software and let’s them decide how to solve them.

   (b) **Consider All Possibilities** tries to be as complete as possible giving the user the ability to choose between every possible value.

3. **Conflict Resolution**: its is computationally more expensive than other strategies but allows to
resolve the conflict as optimally as possible, contrary to the other categories conflict resolution takes into account all the data and meta-data before deciding what strategy to apply and further subdivides itself into two subgroups: Decide and Mediate.

(a) **Cry with the Wolves** the idea behind this strategy is choosing the most common value between the conflicting ones as its probably the correct value, although tie breakers are necessary.

(b) **Roll the Dice** takes into consideration all the values and chooses one at random.

(c) **Meet in the middle** this strategy tries to find a compromise between all the values, alternatively we could minimize the error or take the average of all the values.

(d) **Keep up to Date** meta-data based, it chooses the most recent value based on timestamps, this can be present in the tables as a separate attribute or can be provided by, for example, data lineage facilities Galhardas et al. [2001].
3 Related Work

Contents

3.1 Related Work ................................................. 31
3.2 Duplicate Detection ........................................... 31
3.3 Previous Works On The Portuguese Administrative Census ........................ 33
3.4 Census Related Works ....................................... 35
3.1 Related Work

In this section we will talk about relevant previous works related to the optimization of census using machine learning techniques. To make it clearer I will split it into three sections: Duplicate Detection, which will be further divided into Efficiency and Effectiveness, works related to Census using administrative data performed in other countries.

3.2 Duplicate Detection

The Duplicate Detection problem can be divided into actually finding the existent duplicates (Effectiveness) and into how quickly and cheaply, resource and time wise, we can find them (Efficiency).

3.2.1 Efficiency

Efficiency is obviously important when working with high quantities of data. The blocking step, explained in Section 2.3, is a good example of how reduce the amount of records that need to be paired up and drastically reducing the amount of time and resources required to run this project. Finding the optimal blocking key is, however, a big challenge. Hernández and Stolfo [1998] proposed a technique that consisted in combining the results of several executions of the sorted neighborhood blocking algorithm (Multipass approach) with a different blocking key in each execution.

Michelson and Knoblock [2006] state that the effectiveness of a multi-pass approach depends on the chosen attributes and used methods, according to them the fundamental issue of blocking research is to find what are the best attributes and what are the methods that should be applied to those attributes. To solve these issues they propose a machine learning approach. By automatically learning rules (a conjunction of attributes) using the Sequential Covering Algorithm (SCA) proposed by Mitchell [1997], the effective schema is then evaluated by learning rules that cover a sufficient number of True Positives while minimizing the number of False Positives, using the Reduction Ratio (RR) to quantify how well the current blocking scheme minimizes the number of candidates and the Pair Completeness (PC) to measure the coverage of true positives:

\[
RR = 1 - \frac{C}{N} \quad (3.1) \quad PC = \frac{S_m}{N_m} \quad (3.2)
\]
Where $C$ represents the number of candidate matches, $N$ the size of the cross product between both data sets, $S_m$ the number of true matches in the candidate set and $N_m$ the number of matches in the entire dataset.

Li et al. [2006] propose using a string map function, that will convert the blocking key values to a multidimensional Euclidean space, they then select a threshold and for each pair that is bellow that threshold, preform a multidimensional similarity join that will form clusters of records, the records on these cluster will then be compared with a similarity metric.

### 3.2.2 Effectiveness

Much like efficiency, we should always have effectiveness into consideration, and with that idea in mind TAILOR (Record Linkage Toolbox) was developed by Elfeky et al. [2002]. With TAILOR users can build their own record linkage models by tuning system parameters and by plugging in in-house developed and public domain tools. These authors also proposed three models, one based on supervised learning (Induction Record Linkage Model), one based on unsupervised learning (Clustering Record Linkage Model) and the third one based on both supervised and unsupervised learning (Hybrid Record Linkage Model). Results show that all of these models obtain better accuracy and completeness than the probabilistic model.

According to Christen [2012a], the simplest way to classify two candidate record pairs into matches and non-matches is to sum all similarity scores of the previously compared fields of two different records into a single total similarity metric (the author calls it SimSum) and then compare this values with a similarity threshold to decide into which class each pair belongs. With two classes (Match and Non-Match) only a single threshold is required, if the SimSum values is above the threshold its a Match, if its bellow its a Non-Match. When we have three classes (Match, Non-Match and Possible Match) we need two thresholds (upper and lower), if the SimSum value is bellow the lower threshold is a non-match, if its above the upper threshold, its a match and if its between the thresholds its a potential match that needs to be reviewed by an specialist or a experienced user. Yet if we simply sum all of the fields they all have the same importance therefore if there are fields more important than others we can attribute different weights to each of them.

Weis et al. [2008] presented the DogmatiX prototype, which was designed to detect duplicates in hierarchical XML data. He also tells us how this prototype was applied on a large scale industrial relational database whose main business line is to store and retrieve credit histories of over 60 million individuals.

In short, descriptions for a given candidate type are semi-automatically selected using heuristics and conditions based on an XML Schema (Figure 3.1). To detect duplicates, only the elements that are
closer to the candidate in the schema hierarchy like name or address are considered because they are more useful, while elements that are far from the element are less likely to clearly distinguish person candidates. Furthermore the heuristic selection is still refined by conditions like pruning XML elements based on data type, content model and cardinality.

3.3 Previous Works On The Portuguese Administrative Census

I would like to mention the thesis of Silva [2017] and Velho [2017], which were the basis for my own thesis. Their objective was work was to design a record matching model, that will receive records from different databases and determine if the records are or not duplicates, in other words, find all records that refer to each person.

To offer a deeper understanding about these goals and the type of database they worked with, I will cite Silva [2017]:

"The goal of Statistics Portugal INE is to link/match the records between the databases (those that refer to the same person). Thus, INE is able to apply some residence rules to estimate the number of resident people in Portugal in a determined year with the creation of a database, named the Resident Population Base BPR. BPR contains the records, matched from the different administrative databases, of resident people with the corresponding data."

Using the records provided by INE and other entities, like AT and IISS [1.3], and then try to match them through the different databases using the record matching model. Their by work is divided in 3 parts: the learning phase, the testing phase and monitoring. The process starts by selecting pairs of databases to match with each other. Then it is necessary to perform data cleaning and normalization to
Table 3.1: Example of BPR where NIC, NIF and NISS correspond to different personal identifiers/keys of different databases.

<table>
<thead>
<tr>
<th>NIC</th>
<th>NIF</th>
<th>NISS</th>
<th>Name</th>
<th>Date of Birth</th>
<th>Sex</th>
<th>Birth Country</th>
<th>Postal Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>F34F3F43</td>
<td>-</td>
<td>-</td>
<td>RUI LVA</td>
<td>-</td>
<td>M</td>
<td>Portugal</td>
<td>-</td>
</tr>
<tr>
<td>C34V3V3</td>
<td>ZBBYUBRE</td>
<td>-</td>
<td>MAR CIA</td>
<td>20-12-1995</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>FE24V43</td>
<td>KNUD9S9S</td>
<td>S0EDWC9C</td>
<td>MAN ZES</td>
<td>11-01-89</td>
<td>M</td>
<td>Portugal</td>
<td>1590-741</td>
</tr>
<tr>
<td>-</td>
<td>B65F654533</td>
<td>-</td>
<td>TOM NIO</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>D4D3FF33</td>
<td>-</td>
<td>JOA GAS</td>
<td>-</td>
<td>F</td>
<td>Brazil</td>
<td>-</td>
</tr>
</tbody>
</table>

make sure a single representation for both databases is obtained. Normalization is particularly important, in order to select the fields that are common to both databases, these fields must have the same representations for the data to be consistent.

In the next phase two sets are necessary: True Matches and Unmatched Records. First I select the records that have the same identifier (for e.g: NIF) from both databases, as this identifier is unique, then we pair these records with themselves and select those who have the same blocking key as True positives and those who don’t as True Negatives, both of them will compose the True Matches set. For the blocking criteria the fields First Name + Date of Birth were chosen.

To obtain the Unmatched Records we take the records from both databases that did not have a pair with the same identifier (for e.g: different NIF) but have the the same blocking key as there is a considerable probability of this being some sort of error.

For the training phase we take both True Matches and Unmatched Records sets and calculate the string similarity for every field of all the pairs originating two new sets: True Matches Similarities and the Unmatched Records Similarity set. The values of the True Matches Similarities are then used to create a model that will be used in the next phase.

In the classification phase we simply need to take the model previously created to classify the Unmatched Records Similarity set. To finalize, some queries are applied to the retrieved matches in order to remove non-matches and pairs previously matched by INE to end up exclusively with new matches.

Their monitoring system is split into 4 modules:

1. Data Cleaning and Normalization, this module evaluates the completeness of the database after the data cleaning and normalization using the following Equation:

   \[
   \text{Completeness} = \frac{\text{Number of Non-Null Records}}{\text{Total Number of Records}}
   \]

2. Quality of Blocking. This module allows us to know the number of positives (true matches) or negatives (true non-matches) in the training data for the model. This module also provides the RR of blocking (comparison between the number of comparisons performed with blocking and
the number of comparisons of a cartesian product). The Reduction Ratio formula is given by the following Equation:

\[
RR = 1 - \frac{\text{Number of Comparisons}}{\text{Total Records of Database 1} \times \text{Total Records of Database 2}}
\] (3.4)

3. Quality of the Models. In this module the evaluation metric for each model are calculated. To do so a 2-fold cross validation is performed over the records that were matched through a common key. After applying the Standard Blocking to these matches both positive and negative labeled examples, meaning that half of the set can be used to classify the other half. Then its possible to compare the results with the labeled records using the evaluation metrics described in Section 5.3

4. Quality of the Classification. This module controls the quality of the results, it returns two columns with contradictions and uncertainties.

A contradiction happens when records have different values and the percentage of uncertainty of a certain filed is given by Equation 3.5:

\[
\text{Contradiction} = \frac{\text{Number of records with different values}}{\text{Total Records}}
\] (3.5)

An uncertainty happens when a record has Null values and its percentage is given by Equation 3.6:

\[
\text{Uncertainty} = \frac{\text{Number of records with null values}}{\text{Total Records}}
\] (3.6)

### 3.4 Census Related Works

This section presents some works applied to the problem of Census data.

#### 3.4.1 Big Match

This system was developed on the US Census Bureau record linkage program, by Yancey [2002], to extract subfiles of plausible match records from a very large file, by making only one sequential pass
through the very large file without the necessity of sorting it. In short the system uses two files: a larger file called the Record file and a smaller filled called Memory file that should be small enough to fit in memory. The program then reads the records from the Record file and for each blocking criterion, the program will determine if there are any records in the memory file that have keys that match the key of the record in the Record file. For all the records in the Memory file with a matching key value, a matching comparison weight is computed with the record from the Record file. If any of the comparison weights surpass a certain threshold, the record from the Record file is written into a subfile that corresponds to that blocking criteria.

3.4.2 Beyond 2011

The Beyond 2011 program was initiated in 2011 by the UK Statistics Authority (UKSA)\(^1\) with the objective of looking for different alternatives for a future census in 2021, after the UK Government Treasury Select Committee expressed its concerns about the increasing cost of running the census and the inaccuracy in data that, until then, was only gathered every ten years.

An exercise was performed with 10,000 student records provided by the Higher Education Statistics Agency (HESA)\(^2\) and the National Health Service (NHS)\(^3\) Patient Register. This exercise obtained positive results, by using the method presented in Figure 3.2, it achieved 88.1% compared with a match rate of 89.9% achieved by the gold standard, comprised of a set of perfectly matched records between two datasets, where all of the true Matches were identified and all of the true Non-Matches were left unmatched\(^4\).

In 2014 the UKSA recommended that the England and Wales 2021 census would be conducted mainly through online surveys and the use of administrative data. By January 2015 the UKSA decided to follow the 2014 recommendations by formally closing the Beyond 2011 program and replacing it by the Census Transformation Program (CTP)\(^5\) that would have the main objective of implementing said recommendations.

3.4.3 New Zealand Census

Statistics New Zealand (SNZ)\(^6\) purpose is to improve the quantity and quality of the administrative

---

1. [www.statisticsauthority.gov.uk/](http://www.statisticsauthority.gov.uk/)
2. [www.hesa.ac.uk/](http://www.hesa.ac.uk/)
3. [www.nhs.uk/pages/home.aspx](http://www.nhs.uk/pages/home.aspx)
4. [www.tinyurl.com/y76bpxsq](http://www.tinyurl.com/y76bpxsq)
5. [www.ons.gov.uk/census/censustransformationprogramme](http://www.ons.gov.uk/census/censustransformationprogramme)
data provided to New Zealand by 2030. To improve the quality of the data it is especially important to
design, test, and implement more modern and targeted collection approaches and new communication
approaches to support an online census. In order to make a reduction in the costs of supplementary
variables the use of administrative data was adopted. The data needs to be cleaned and normalized,
and, due to privacy concerns the personal identifiers were also encrypted. Then Standard Blocking is
used with the field Date of Birth as a blocking criteria to reduce the number of comparisons and errors.
Other blocking techniques were also utilized to match pairs that the Standard Blocking wasn’t able to
handle, like Soundex on the First and Last names.

The method Fellegi-Sunter I. P. Fellegi [1969] was used to calculate the Agreement and Disagree-
ment weights (AW and DW respectively) to find possible duplicates.

\[ AW(m, u) = \log_2 \frac{m}{u} \quad \text{(3.7)} \quad DW(m, u) = \log_2 \frac{1 - m}{1 - u} \quad \text{(3.8)} \]

With \( m \) representing reliability (Equation 3.7) and \( u \) representing commonality (Equation 3.8) and
can be calculated through the following formulas:

\[ m = \text{Prob}(\text{Two values agree } | \text{ The records are a match}) \quad \text{(3.9)} \]
\[ u = \text{Prob}(\text{Two values agree} \mid \text{The records are not a match}) \] (3.10)

In order for a pair to be considered a match, each field was evaluated by a metric that decides if the values that both records have in that particular field agree or disagree. After all fields are evaluated, their scores are summed into a final score. The final score is then compared to a threshold, only if the score is higher than the threshold will the pair be considered a match.
4
Proposal

Contents

4.1 Proposal ......................................................... 41
4.1 Proposal

My objective was to develop on the project originally created by Silva [2017] and Velho [2017] represented in Figure 4.1, more specifically on the Blocking, Training and Classification Component.

Figure 4.1: Record Linkage System Architecture.

I will begin by making a brief description of my colleagues’ approach to the problem and then following up with my own proposal to improve upon their work.

To achieve an acceptable level of reliability, a cross-reference between databases is necessary to make sure that there are no mistakes and each record belongs exclusively to one person.

Then, to make the matching process computationally feasible Silva [2017] and Velho [2017] used the standard blocking technique to group similar records using a blocking key. Yet, by using this technique with a specific blocking key, some potential matches may be excluded if there are errors on the chosen fields (First Name, Birth Date).

To compare the records within these groups, I used the Edit Distance algorithm, given its simplicity. The feature extraction chosen method was also very simplistic: they calculate the string distance between the different fields, and use these distances as features to train the model that will classify the unmatched records using the Logistic Regression Classifier.

After obtaining a dataset with record pairs classified by the previously created model, this model takes into consideration all of the similarities between every field for each record pair, and calculates the probability of that pair being a match. If the probability is above 50%, it is considered a match, otherwise a non-match, and the pairs classified as non-matches are discarded, as we are not interested in them. Afterwards, the process checks if the obtained records exist in the BPR, because we only want the new
But there is still an issue that needs to be addressed, this process is not flawless, as we may have pairs that the model considered to be a match and do not exist in the BPR, but in reality that is not the case. Let’s look at Table 4.1 as an example.

<table>
<thead>
<tr>
<th>Database A</th>
<th>Database B</th>
<th>Probability of Being a Match (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record 1</td>
<td>Record 2</td>
<td>98%</td>
</tr>
<tr>
<td>Record 1</td>
<td>Record 4</td>
<td>67%</td>
</tr>
<tr>
<td>Record 1</td>
<td>Record 5</td>
<td>58%</td>
</tr>
</tbody>
</table>

Table 4.1: Possible Matches not yet present in BPR.

As we can see, the model considers that the Record 1 from Database A matches the Records 2, 4 and 5 from Database B, but in reality only one of these matches can be considered true because each record can only have a single match between a specific pair of databases. To solve this problem, the process has a final step where it will only select the match with the highest probability.

Moving on to my own proposal, the following list presents what would be my main objectives to improve upon this work:

• Replicate the matching with the 2015 data to ensure that the previous code was working and to get familiarized with the project.

• Replicate the matching with the 2016 data to ensure it was compatible with the project and to create the necessary tables to move to the next stage.

• Attempt to reduce the process’ running time by testing different blocking keys.

• Address the overfitting problem by performing cross-validation with a different number of folds.

• Test a new string matching algorithm apart from the previously used Edit Distance, like Jaccard or Jaro-Winkler.

• Test a new machine learning algorithm apart from the previously used Logistic Regression, like Support Vector Machines or Decision Trees.

• Measure the improvements obtained through the evaluation metrics mentioned in section 5.3.

To accomplish these goals, I would be intervening mainly on the colored components from Figure 4.1, but due to several issues with the previous code, I eventually work in all of them.
5

Experiments

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Development Environment</td>
<td>45</td>
</tr>
<tr>
<td>5.2 Data Characterization</td>
<td>45</td>
</tr>
<tr>
<td>5.3 Evaluation</td>
<td>45</td>
</tr>
</tbody>
</table>
In this chapter, I overview the data used for this work and the methods used to evaluate the quality of the obtained matches.

5.1 Development Environment

The databases are stored in an Oracle Server and we perform queries through the graphical tool SQL Developer. We access SQL Developer through a Linux server (Debian) with 32GB of RAM and 2 CPU’s Intel(R) Xeon(R) CPU X5680 3.33GHz. The programming language I used for training the model and classifying the records was mainly Python. In addition, I used the Logistic Regression, Decision Trees, SVM’s, Naive Bayes and Random Forest from the Scikit Learn library\(^1\). To create a connection between my workspace and the server I used Putty and X-Ming\(^2\) to provide me a graphical interface that allowed me to navigate the project folders in a file explorer. For the calculation of the string similarity I used the Levenshtein, Hamming, Jaccard, Jaro-Winkler and Dice algorithms from the Textdistance library\(^3\). To load the records from a CSV file (downloaded from SQL Developer) I used the Pandas Dataframe\(^4\).

5.2 Data Characterization

As described in Section 1.3, the data was provided to me by INE, but a big portion of this data is granted to INE by several external entities and is stored in separated databases. In Table 5.1 we can see which fields are present in each database.

5.3 Evaluation

After cleaning the data and classifying it, we need to evaluate the quality of the returned matches. In a project that works with sensitive information, it’s imperative that data coming from the sources has a minimum level of quality and reliability.

When declaring two records as either a Match or Non-Match, there are four different types of possible results: True Positives(TP), False Positives(FP), True Negatives(TN), False Negative(FN). In Table 5.2 we can see a schematic representation, making it easier to understand. \(P\) represents our prediction, \(C\)

\(^1\)www.scikit-learn.org/stable/
\(^2\)www.straightrunning.com/XmingNotes/
\(^3\)www.pypi.org/project/textdistance/
\(^4\)www.pandas.pydata.org/
<table>
<thead>
<tr>
<th>Attribute</th>
<th>AT</th>
<th>BDIC</th>
<th>CGA</th>
<th>EDUC</th>
<th>IEFP</th>
<th>IISS</th>
<th>SEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIC</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NIF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>FIRST NAME</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LAST NAME</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SEX</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BIRTH YEAR</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BIRTH MONTH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BIRTH DAY</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>MARITAL STATUS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NATURALITY DISTRICT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NATURALITY COUNTY</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NATURALITY PARISH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>NATURALITY CODE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RESIDENCE DISTRICT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RESIDENCE COUNTY</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RESIDENCE PARISH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ZIP CODE 4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ZIP CODE 3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LOCALITY RESIDENCE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>RESIDENCE CODE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 5.1: This table shows which fields are present in each database.

represent their real classification, $M$ represents a Match and $N$ represents a Non-Match.

<table>
<thead>
<tr>
<th></th>
<th>$P = M$</th>
<th>$P = N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C = M$</td>
<td>TP</td>
<td>FN</td>
</tr>
<tr>
<td>$C = N$</td>
<td>FP</td>
<td>TN</td>
</tr>
</tbody>
</table>

Table 5.2: Classification possible outcomes.

True Positives represent the pairs that we predicted to be a Match and are actually a Match, False Positives the pairs we predicted to be Matches but are in fact a Non-Match, False Negatives stand for the pairs that we predicted to be Non-Matches but are Matches and finally, True Positives represent the pairs that we predicted to be Non-Matches and are actually Non-Matches. Using these definitions, I will now present the most commonly used classification evaluation measures.

### 5.3.1 Evaluation Metrics

In essence, **Precision** represents the number of actual matches (True Matches) amongst all the returned matches. To capture the large number of negative examples on the algorithm’s performance, the **Precision** will compare the false positives to true positives Davis and Goadrich [2006].
\[ \text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}} \quad (5.1) \]

Recall represents the number of returned matches from all the actual matches and it is presented in Equation 5.2.

\[ \text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}} \quad (5.2) \]

Accuracy is the ratio between the True results (True Positives + True Negatives) and everything else returned (True Positive + True Negative + False Positive + False Negative). This ratio is given by in Equation 5.3

\[ \text{Accuracy} = \frac{\text{TruePositive} + \text{TrueNegative}}{\text{TruePositive} + \text{FalsePositive} + \text{TrueNegative} + \text{FalseNegative}} \quad (5.3) \]
6

Results
6.1 Results

In this section I will present the results obtained during the six months I’ve worked with INE.

6.2 Bug Fixing

When I started working on this project I encountered some of the bugs that were caused by out-of-date libraries, this problem was quickly fixed by updating these libraries, yet some bugs related to the code remained and required me to spend some time fixing them.

After all the python related problems were fixed, a new issue related to the SQL Oracle appeared: "Unable to extend table space". This happened due to the lack of space to create some of the bigger tables, but not the smaller ones, which was a bit confusing until I realized that the size of the tables I was trying to create could range from a few Megabytes to 10 Gigabytes or more. This issue was eventually solved by increasing the Table Space size.

6.3 Comparing Results

In the following, Table 6.1 I will present the figures that represent the number of new matches, referenced in Section 4.1 for the year 2015.

<table>
<thead>
<tr>
<th></th>
<th>Sampaio &amp; Silva Results</th>
<th>My Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDIC AT</td>
<td>244.903</td>
<td>284.985</td>
</tr>
<tr>
<td>BDIC IISS</td>
<td>47.836</td>
<td>55.631</td>
</tr>
<tr>
<td>BDIC EDUC</td>
<td>51.138</td>
<td>36.184</td>
</tr>
<tr>
<td>BDIC IEFP</td>
<td>11.974</td>
<td>21.311</td>
</tr>
<tr>
<td>BDIC CGA</td>
<td>60.545</td>
<td>60.160</td>
</tr>
<tr>
<td>IISS SEF</td>
<td>30.120</td>
<td>29.950</td>
</tr>
<tr>
<td>AT SEF</td>
<td>52.177</td>
<td>52.142</td>
</tr>
<tr>
<td>SEF EDUC</td>
<td>12.796</td>
<td>11.457</td>
</tr>
</tbody>
</table>

Table 6.1: Differences between the amount of new matches found by Silva [2017] and Velho [2017] and the amount of new matches I found for the year 2015.

As it is shown in Table 6.1, some of the results are very similar to the ones obtained by my colleagues, while others are considerably different. This happened due to the fact that I used SQL scripts created
by myself (the scripts used by Silva [2017] and Velho [2017] were no longer available), which produced different data and in turn, different results as expected.

I then tested the possibility of using different blocking keys as an alternative to the one currently in use (First Name, Year of Birth, Month of Birth, Day of Birth), with the following combinations:

- Last Name, Year of Birth, Month of Birth, Day of Birth.
- Last Name, Postal Code.
- First Name, Last Name.

Unfortunately, this would either generate very few candidates for classification, due to a very strict key or many null entries, for example the postal code field, meaning that only a few thousands of records could be used, or it would generate too many candidates due to a less strict key, which would make the process extremely time consuming and simply unfeasible.

Similarly, when I tested the training component with a different number of folds, it would simply increase the time necessary to finish the process, without any noteworthy impact on the results, so I decided I would continue to use the 2-fold approach.

Afterwards, I started creating the tables with the data from the year 2016. Given that I obtained different tables than Silva [2017] and Velho [2017] using the 2015 data, it wouldn’t make sense to compare the tables created from the 2016 data, using my scripts, to the tables from 2015, that Silva [2017] and Velho [2017] used, and expect an improvement, because they are not equivalent. Therefore, from this point forward, I used my 2015 results as a baseline.

<table>
<thead>
<tr>
<th>2015 Results</th>
<th>2016 Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDIC AT</td>
<td>284.985</td>
</tr>
<tr>
<td>BDIC IISS</td>
<td>55.631</td>
</tr>
<tr>
<td>BDIC EDUC</td>
<td>36.184</td>
</tr>
<tr>
<td>BDIC IEFP</td>
<td>21.311</td>
</tr>
<tr>
<td>BDIC CGA</td>
<td>60.160</td>
</tr>
<tr>
<td>IISS SEF</td>
<td>29.950</td>
</tr>
<tr>
<td>AT SEF</td>
<td>52.142</td>
</tr>
<tr>
<td>SEF EDUC</td>
<td>11.457</td>
</tr>
</tbody>
</table>

Table 6.2: Differences between the amount of new matches found with data from 2015 and new matches found with data from 2016.

As expected, the amount of new matches found with data from 2016 are lower than the ones previously achieved. This is due to the fact that the new records found by Silva [2017] and Velho [2017] were incorporated into the BPR matrix. Therefore, the majority of the new records found by me already exist in BPR and are no longer considered “new”. Yet, there are some exceptions where I obtain higher
results than my colleagues previously did. These exceptions happened because the SQL scripts used by me had some differences (as I was never able to fully replicate the code my colleagues used), leading to different results.

6.4 String Matching and Classification Algorithms

Originally, Silva [2017] and Velho [2017] used simple algorithms for the matching of strings in the existing records (ex: Name, Address) and for the classification of record pairs the chosen algorithms were Edit Distance and Logistic Regression, due to their simplicity. Yet, more complex and interesting algorithms exist, and one of the goals for this thesis was to test these alternatives and find out which work better.

For the string similarity calculation, I tested the following algorithms: Jaccard, Jaro-Winkler, Hamming and Dice. The results obtained are shown in Table 6.3 that contains the accuracy obtained by each algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Levenshtein</th>
<th>Jaccard</th>
<th>Hamming</th>
<th>Dice</th>
<th>Jaro-Winkler</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEF EDUC</td>
<td>0.9129</td>
<td>0.9132</td>
<td>0.9105</td>
<td>0.9145</td>
<td>0.9105</td>
</tr>
<tr>
<td>AT SEF</td>
<td>0.9895</td>
<td>0.9928</td>
<td>0.9882</td>
<td>0.9853</td>
<td>0.9781</td>
</tr>
<tr>
<td>IISS SEF</td>
<td>0.9799</td>
<td>0.9836</td>
<td>0.9793</td>
<td>0.9772</td>
<td>0.9749</td>
</tr>
<tr>
<td>BDIC CGA</td>
<td>0.9761</td>
<td>0.9708</td>
<td>0.9757</td>
<td>0.9690</td>
<td>0.9710</td>
</tr>
<tr>
<td>BDIC IEFP</td>
<td>0.9858</td>
<td>0.9881</td>
<td>0.9857</td>
<td>0.9841</td>
<td>0.9841</td>
</tr>
<tr>
<td>BDIC EDUC</td>
<td>0.9813</td>
<td>0.9839</td>
<td>0.9815</td>
<td>0.9769</td>
<td>0.9776</td>
</tr>
<tr>
<td>BDIC IISS</td>
<td>0.9984</td>
<td>0.9989</td>
<td>0.9984</td>
<td>0.9759</td>
<td>0.9958</td>
</tr>
<tr>
<td>BDIC AT</td>
<td>0.9991</td>
<td>0.9990</td>
<td>0.9991</td>
<td>0.9851</td>
<td>0.9950</td>
</tr>
</tbody>
</table>

Table 6.3: Accuracy for the string matching algorithms (using Log. Regression for classification).

It is important to note that these results tell us how accurate a certain classification algorithm is while using the features (in this case the similarities) provided by the string matching algorithms. The chosen classification algorithm was the Logistic Regression, again, for its simplicity, as what I was really trying to understand at this stage was what string matching algorithm provides the best features. Ultimately, the Jaccard algorithm offered slightly better results than the rest.

To perform the classification of records, the following algorithms were tested: Decision Trees, Support Vector Machines, Naive Bayes and Random Forest. The results are presented in Table 6.4.

As we can see in Table 6.4, there is no clear margin between the results, as they all have a pretty similar accuracy. However, the combination of Jaccard for string matching and Decision Trees for classification does have a slightly higher accuracy for the majority of the database pairings.

In Table 6.5, we can see the results from running the process with both the previously used algorithms
### Table 6.4: Accuracy for the Classification Algorithms (using Jaccard distance for string matching).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>SEF EDUC</th>
<th>AT SEF</th>
<th>IISS SEF</th>
<th>BDIC CGA</th>
<th>BDIC IEFP</th>
<th>BDIC EDUC</th>
<th>BDIC IISS</th>
<th>BDIC AT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log. Regression</td>
<td>0.9132</td>
<td>0.9251</td>
<td>0.9153</td>
<td>0.9068</td>
<td>0.8988</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dec. Trees</td>
<td>0.9928</td>
<td>0.9927</td>
<td>0.9927</td>
<td>0.9904</td>
<td>0.9887</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>0.9036</td>
<td>0.9850</td>
<td>0.9790</td>
<td>0.9608</td>
<td>0.9851</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.9708</td>
<td>0.9758</td>
<td>0.9883</td>
<td>0.9516</td>
<td>0.9516</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.9881</td>
<td>0.9882</td>
<td>0.9883</td>
<td>0.9812</td>
<td>0.9815</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BDIC EDUC</td>
<td>0.9839</td>
<td>0.9856</td>
<td>0.9831</td>
<td>0.9823</td>
<td>0.9788</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BDIC IISS</td>
<td>0.9989</td>
<td>0.9992</td>
<td>0.9987</td>
<td>0.9940</td>
<td>0.9967</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BDIC AT</td>
<td>0.9990</td>
<td>0.9993</td>
<td>0.9990</td>
<td>0.9979</td>
<td>0.9975</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 6.5: Comparison between the results from previous solution (with Levenshtein - Logistic Regression) and the new solution (with Jaccard - Decision Trees).

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Unmatched Records (a)</th>
<th>Previous Solution</th>
<th>New Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PM (b)</td>
<td>MIM (c)</td>
<td>NM (d)</td>
</tr>
<tr>
<td>AT</td>
<td>4.041.101</td>
<td>663.642</td>
<td>598.940</td>
</tr>
<tr>
<td>BDIC</td>
<td>1125.575</td>
<td>43.307</td>
<td>36.564</td>
</tr>
<tr>
<td>EDUC</td>
<td>36.564</td>
<td>255.476</td>
<td>228.034</td>
</tr>
<tr>
<td>BDIC</td>
<td>13.053.206</td>
<td>34.582</td>
<td>25.780</td>
</tr>
<tr>
<td>IEFP</td>
<td>196.135</td>
<td>130.114</td>
<td>71.510</td>
</tr>
</tbody>
</table>

(a) Number of records to match per database.
(b) Positive Matches.
(c) Matches that already exist in the matrix.
(d) New Matches.

and with the newly used ones, there is an increase in the number of new matches found (matches that did not previously exist in BPR) for the pairs BDIC-AT, BDIC-IISS, BDIC-CGA and SEF-EDUC yet, for BDIC-EDUC, BDIC-IEFP, IISS-SEF and AT-SEF, the amount of new matches found is lower. Given the fact that the new solution (using Jaccard and Decision Trees) obtained a slight increase in accuracy, a higher amount of new matches should be expected. Consequently, these results need to be further examined by INE to determine the problem. There are a few plausible causes that may explain such behaviour, for example: when I calculate the New Matches for Previous Solution, the new found matches are not added to BPR (adding new matches to BPR is out of this project scope). Therefore, many “New” matches found by both solutions may overlap.

To put it simply, let’s take the pair **BDIC AT** as an example, the previous solution obtained 9.055 new matches and the new solution obtained 9.208, this could mean that 9.055 matches are common to both solutions which would result in a 153 matches difference. But the two solutions might not have any new
match in common, which would mean an approximate 9,000 matches difference between them. Could this imply that one of the solutions is completely wrong? And, if that is indeed the case, which solution is correct?

6.5 Interface Improvement

After implementing these new algorithms one by one and testing them, I thought it wouldn’t be practical to change the code each time I wanted to test a different combination of a string similarity and a classification algorithm, so I decided to update the menu previously created by Silva [2017] and Velho [2017] and give the users the ability to choose the implemented algorithms and test them as they see fit.

While running, the process would also print the number of the pair being classified, which would quickly clutter the console. This was problematic while debugging and in general was simply inconvenient. Therefore, I created a simple Python script that would print the number once and it would change dynamically as the pairs were classified.

By INE’s requirement, a few tweaks to the way inputs were taken were also made, amongst others, the ability to take the names of the databases in both lower and capital letters which does make the input process easier and less prone to mistakes.

6.6 Monitoring

During the development of this project, I used a straightforward approach to get back my results after running the entire process, specially the string matching and classification phases. I would simply use a library to print the data do the console at the end of the running cycle and would save those results so I wouldn’t have to run it again, with the same parameters. Since the beginning, my intention was to re-use the monitoring section from the previous project. However, some components, like the blocking monitoring component, required some tweaks and others needed to be finished, such as the classification monitoring component. In this section, I will talk about some of these components and show what kind of information they provide.

The first component is responsible for monitoring the normalization phase: for each of the normalized tables we get information about the total number of records in that dataset, and for each existing field we get the null count, distinct values count and the completeness percentage [3.3].
### Table 6.6: Number of BDIC Records from 2016.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Completeness</th>
<th>Null Count</th>
<th>Distinct Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROV</td>
<td>100%</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>ID</td>
<td>100%</td>
<td>0</td>
<td>1386822</td>
</tr>
<tr>
<td>FIRST NAME</td>
<td>100%</td>
<td>0</td>
<td>5111</td>
</tr>
<tr>
<td>LAST NAME</td>
<td>99%</td>
<td>140</td>
<td>5307</td>
</tr>
<tr>
<td>SEX</td>
<td>100%</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>BIRTH YEAR</td>
<td>100%</td>
<td>0</td>
<td>138</td>
</tr>
<tr>
<td>BIRTH MONTH</td>
<td>99%</td>
<td>54</td>
<td>14</td>
</tr>
<tr>
<td>BIRTH DAY</td>
<td>99%</td>
<td>54</td>
<td>33</td>
</tr>
<tr>
<td>MARITAL STATUS</td>
<td>100%</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>NATURALITY DISTRICT</td>
<td>100%</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>NATURALITY COUNTY</td>
<td>100%</td>
<td>0</td>
<td>327</td>
</tr>
<tr>
<td>NATURALITY PARISH</td>
<td>100%</td>
<td>0</td>
<td>3493</td>
</tr>
<tr>
<td>NATIONALITY CODE</td>
<td>100%</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>RESIDENCE DISTRICT</td>
<td>100%</td>
<td>0</td>
<td>31</td>
</tr>
<tr>
<td>RESIDENCE COUNTY</td>
<td>100%</td>
<td>0</td>
<td>310</td>
</tr>
<tr>
<td>RESIDENCE PARISH</td>
<td>100%</td>
<td>0</td>
<td>3356</td>
</tr>
<tr>
<td>ZIP CODE 4</td>
<td>68%</td>
<td>4403269</td>
<td>639</td>
</tr>
<tr>
<td>ZIP CODE 3</td>
<td>68%</td>
<td>4407423</td>
<td>980</td>
</tr>
<tr>
<td>LOCALITY RESIDENCE</td>
<td>78%</td>
<td>3010182</td>
<td>294919</td>
</tr>
<tr>
<td>RESIDENCE CODE</td>
<td>100%</td>
<td>0</td>
<td>197</td>
</tr>
</tbody>
</table>

*Table 6.7: Analysis on Completeness, Number of Null Records and Distant Values for each Field in BDIC 2016.*

Blocking was used to reduce the amount of comparisons between two databases [2.3]. This component is responsible for displaying the number of true positives and true negatives used to create the classification model, as well as the number of total possible matches between the databases and the reduction ratio obtained for both of them.

<table>
<thead>
<tr>
<th>Analyzed Category</th>
<th>Total Records</th>
<th>Reduction Ratio using Blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive Pairs</td>
<td>794538</td>
<td>99.99998%</td>
</tr>
<tr>
<td>Negative Pairs</td>
<td>4441932</td>
<td></td>
</tr>
<tr>
<td>Candidate Pairs</td>
<td>5329960</td>
<td>99.99996%</td>
</tr>
</tbody>
</table>

*Table 6.8: Reduction Ratio using blocking for the pair BDIC-CGA.*
After having determined that the best choice for algorithms would be Jaccard - Decision Trees, I generated a new model for the classification and measured the Precision and Recall for both the true positive and true negative matches.

<table>
<thead>
<tr>
<th>Models</th>
<th>Precision(%)</th>
<th>Recall(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>SEF EDUC</td>
<td>96%</td>
<td>86%</td>
</tr>
<tr>
<td>AT SEF</td>
<td>95%</td>
<td>100%</td>
</tr>
<tr>
<td>IISS SEF</td>
<td>98%</td>
<td>98%</td>
</tr>
<tr>
<td>BDIC CGA</td>
<td>99%</td>
<td>91%</td>
</tr>
<tr>
<td>BDIC IEFP</td>
<td>99%</td>
<td>99%</td>
</tr>
<tr>
<td>BDIC EDUC</td>
<td>99%</td>
<td>97%</td>
</tr>
<tr>
<td>BDIC IISS</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>BDIC AT</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Table 6.9:** Quality analysis of the classification model.

The results of Table 6.9 give us a prediction on how the model will behave with the training data, but it does not give much information about how it will actually behave with the unmatched records. Therefore, it was necessary to have some metrics to measure the quality of the final results.

<table>
<thead>
<tr>
<th>Database</th>
<th>Year</th>
<th>Number of Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>BDIC CGA</td>
<td>2016</td>
<td>45329</td>
</tr>
</tbody>
</table>

**Table 6.10:** Number of new matches obtain by pairing BDIC with CGA from 2016.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Contradiction (Records)</th>
<th>Contradiction (%)</th>
<th>Uncertainty (Records)</th>
<th>Uncertainty (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRST NAME</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LAST NAME</td>
<td>1385</td>
<td>3.055</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SEX</td>
<td>22</td>
<td>0.0485</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BIRTH YEAR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BIRTH MONTH</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>BIRTH DAY</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MARITAL STATUS</td>
<td>0</td>
<td>0</td>
<td>45329</td>
<td>100</td>
</tr>
<tr>
<td>NATURALITY DISTRICT</td>
<td>0</td>
<td>0</td>
<td>45329</td>
<td>100</td>
</tr>
<tr>
<td>NATURALITY COUNTY</td>
<td>0</td>
<td>0</td>
<td>45329</td>
<td>100</td>
</tr>
<tr>
<td>NATURALITY PARISH</td>
<td>0</td>
<td>0</td>
<td>45329</td>
<td>100</td>
</tr>
<tr>
<td>NATIONALITY CODE</td>
<td>0</td>
<td>0</td>
<td>45329</td>
<td>100</td>
</tr>
<tr>
<td>RESIDENCE DISTRICT</td>
<td>21196</td>
<td>46.760</td>
<td>16216</td>
<td>35.774</td>
</tr>
<tr>
<td>RESIDENCE COUNTY</td>
<td>35103</td>
<td>77.440</td>
<td>5605</td>
<td>12.365</td>
</tr>
<tr>
<td>RESIDENCE PARISH</td>
<td>0</td>
<td>0</td>
<td>45329</td>
<td>100</td>
</tr>
<tr>
<td>ZIP CODE 4</td>
<td>3608</td>
<td>7.959</td>
<td>40144</td>
<td>88.561</td>
</tr>
<tr>
<td>ZIP CODE 3</td>
<td>4150</td>
<td>9.155</td>
<td>40144</td>
<td>88.568</td>
</tr>
<tr>
<td>RESIDENCE CODE</td>
<td>3739</td>
<td>8.249</td>
<td>40144</td>
<td>88.556</td>
</tr>
</tbody>
</table>

**Table 6.11:** Analysis on the Contradictions and Uncertainties for each field on the New Matches for the BDIC-CGA pair.

The last monitoring component retrieves information to analyze the quality of the new matches found. In Table 6.11 are the results of BDIC and CGA, where we have some metrics for each field. In the first and second columns, we have the number and percentage of contradictions between two records for
a specific field. A contradiction happens when the values of the records are different [Equation3.5]. In the third and fourth columns, we got number and percentage of uncertainties between two records for a specific field, which represent the number of records that have a null value [Equation3.6].

6.7 Summary

After fixing some of the bugs from the previous project and having to generate my own Tables that were as close as the ones created by Silva and Velho as I could get, I began comparing the number of new records found, first using the same algorithms used by Silva and Velho and using data from 2015. Unfortunately I had to generate my tables using my own SQL scripts because the ones used previously no longer existed. This implied that I had a different set of data to work with, and even though I tried to replicate the lost tables as closely as possible, some differences remained, which would have an impact on the results. For the pairs BDIC-CGA, IISS-SEF, AT-SEF, SEF-EDUC I obtained a higher amount of new matches, and for the pairs BDIC-AT, BDIC-IISS, BDIC-EDUC, BDIC-IEFP the amount of new matches found is lower.

When I was finished with the 2015 data, I tested alternative blocking keys and a different number of folds for cross validating, but soon realized these alternatives made the process’ runtime unacceptably long.

I then moved to the 2016 data and, after testing several algorithms, Jaccard and Decision Trees emerged as the best choice by obtaining a higher accuracy for the majority of the database pairings.

Despite this fact, using Jaccard and Decision Trees only allowed me to find a higher amount of new records in 4 out of 8 database pairings (BDIC-AT, BDIC-IISS, BDIC-CGA and SEF-EDUC) meaning that, in order to reach an informed conclusion, further examinations are required from INE in order to validate the new matches.
7 Conclusion and Future Work

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 Conclusion</td>
<td>61</td>
</tr>
<tr>
<td>7.2 Future Work</td>
<td>62</td>
</tr>
</tbody>
</table>
7.1 Conclusion

The objective of this work was to obtain a model capable of taking records from several sources and preform a multi-database cross reference in order to better detect duplicates, errors and inconsistencies in a computational acceptable time. In this section, I will describe the decisions I’ve made along the way whilst providing my opinion about it and making recommendations for the future.

First, I needed to replicate the results obtained in the previous work with the 2015 data to make sure everything was set to move to the 2016 data. This was only partially possible, given that some critical elements from the previous work were missing. After achieving results as similar to the previous work as possible, it was time to test the 2016 data, which as expected, yielded lower results in terms of new matches found in BPR compared to 2015, given that many of them had already been found and integrated to BPR.

Once all the data from 2015 and 2016 had been classified, I moved to the next step: testing the same process with new algorithms, with the goal of possibly finding new matches and improving performance.

Four new string matching algorithms (Jaccard, Jaro-Winkler, Hamming and Dice) and four new classification algorithms (Support Vector Machines, Decision Trees, Naive Bayes and Random Forest) were implemented, along with a few interface improvements to make it easier for the user to select the intended options. The reason behind the choice of algorithms was straightforward: I wanted to use well known algorithms, as it made it easier for people with some experience in the field to know what was happening behind the classification phase, and because it was also easier to find single libraries that included all of the required algorithms.

After having every algorithm implemented, I decided that I should test the process with the 2016 data, as this is where new matches where more likely to be found first, to calculate the accuracy of every algorithm implemented (both similarity and classification algorithms) to determine the best option.

The results showed that the Levenshtein algorithm and the Decision Trees algorithm for the string similarity and classification components respectively were the best choice for most table pairings. Jaccard distance (using Log. Regression for the classification) managed to achieve an higher accuracy in 75% of the database pairs, while the Decision Trees classifier (using Jaccard for string matching) achieved a higher accuracy in 50% of the database pairs.

Despite having achieved better accuracy with the new algorithms, this doesn’t always translate to more records found. Some of the smaller discrepancies need to be further examined by INE to determine the causes.

By the end of the project, I documented everything I did and created a read-me Annex A to ensure the next person to work in this project makes the transition as smoothly as possible.
7.2 Future Work

I will dedicate this section to talk about further improvements and the future possibilities for this project. One way to optimize this process is by having more than a single blocking method. This will allow the reduction in the exclusion of records from the matching process due to inconsistencies, for example, for this work I use the Last Name and the Birth Date (day, month, year) for the blocking process however, when I try to match two records, even if they are a positive match and in fact represent the same person but one of the records has a error in one of this fields, this match will be excluded.

There are also plenty of alternative algorithms for both the string matching and classification phase that could provide better results. This process is not restricted to existing ones, given that new algorithms might be developed even as I write this thesis. With this issue in mind, I made the implementation of new algorithms as modular and simple as possible, and the fact that this project is written using Python is also helpful, given the amount of exiting libraries.

The interface could also be further developed to present the information in a more user friendly format, generating graphics and statistics. Another aspect that could be improved is the ability to take the names of the databases in any order (BDIC_AT or AT_BDIC) because, currently, the project will only recognize one of them, and this would be particularly useful for the INE staff, given the amount of data they have to work with, making the work faster and less prone to errors.

INE also has plans to adapt the process to match addresses in the future, given that the address field is usually where more errors occur. By using this process, INE hopes to find duplicates and correct mistakes.
Bibliography


E. N. O. Gurneet Kaur. A review article on the nayve bayes classifier with various smoothing techniques. 2014. URL https://pdfs.semanticscholar.org/1c41/b7b724e1245201c895160fb46cdd84dca809.pdf.


Census Optimization Using Machine Learning

Required Libraries

To make the project work the following Libraries are required:

- cx_Oracle
- nltk
- sklearn
- numpy
- textdistance
- pandas

Instructions

To run the project take the following steps:

1. Create normalized tables:

   - Go to the queries folder, then select a new folder that corresponds to the tables you want to pair up.
   - Run the script 1_"name_of_table".sql for each of the tables you want to normalize (for example: 1_AT).
   - As an alternative you can go to the prototype folder ans run the script_main_test.py and select the option normalization (2 times, one for each table).

2. Training:
• In the same folder (corresponding to the tables you want to pair up) run the script 3."name_of_pair".sql 
(ex:3.cand_pos_bdic_at) (script 2 is included in script 3) or alternatively go to the prototype folder 
and run the file script_main_test.py and select training.

3. Classification:

• In the same folder run the script 4."table1".norm_nm."table2".sql, and then the script 
5.cand_nm."table1","table2","year".sql.

• In Oracle extract the tables cand_nm."table1",table2"."year" and cand_pos."table1",table2"."year", 
while extracting the table cand_nm."table1",table2"."year", extract only the columns that come after 
nome_3pri excluding rec_id, id_null, identificador_tabela1 e identificador_tabela2), 
table cand_pos."table1",table2"."year", extract only the columns that come after nome_3pri excluding rec_id.

• Go to training folder and run the file similarities.py ( 2 times, one for each table ) choose the 
option non-classified for the cand_nm."table1",table2"."year" table and the option classified for the 
cand_pos."table1",table2"."year" table.

• Run the file train_model.py and type the year and the names for the pair for which you want to 
generate a model.

• Run the file match_records.py and type the year and the names for the pair you want to train.

• Use Oracle to import the resulting matches."tabela1",tabela2"."ano".csv file ( you will need to 
change the size of the size/precision field corresponding to the column rec_id to 9.

• Return to the folder "queries" and run the script 6.class."table1",table2"."year".sql.

• Run the script 7.class."table1","table2",n_matriz","year".sql.

• Run the script 8.class.prob_max."table1","table2","year".sql.

• Run the scripts contradicao."table1","table2","year".sql and incerteza."table1","table2","year".sql 
(required to perform monitoring).

4. Monitoring:

• Go to the monitoring folder and run the file monitorização.py.

• To obtain information about table normalization select option 1.

• To obtain information about record percentage select option 2.

• To obtain information about record classification select option 3.

• To obtain information about contradiction and uncertainty percentage select option 4.