Lucas André do Rego
Furtado Ferreira

Consolidation and Extraction
of Address Information with Intelligence

1 2 9 0
UNIVERSIDADE D
COIMBRA

Lucas André do Rego Furtado Ferreira

Consolidation and Extraction of Address Information with Intelligence

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Prof. Catarina Silva and Prof. Nuno Antunes and presented to the Faculty of Science and Technology / Department of Informatics Engineering.

January 2023

Faculty of Sciences and Technology

Department of Informatics Engineering

Consolidation and Extraction of Address Information with Intelligence

Lucas André do Rego Furtado Ferreira

Dissertation in the context of the Master in Informatics Engineering, Specialization in Software Engineering, advised by Prof. Catarina Silva and Prof. Nuno Antunes and presented to the Faculty of Science and Technology / Department of Informatics Engineering.

September 2022

1 2 9 0

UNIVERSIDADE D
COIMBRA

This page is intentionally left blank.

Acknowledgements

I would like to thank my supervisors Nuno Antunes and Catarina Silva for their guidance and constant support at critical times in the making of this thesis, which many times made me consider giving up. I also greatly appreciate the many revisions they provided to this document. A special thanks goes to R'DEI, my dear friends who have accompanied me from undergraduate to master, and who helped me in structuring some of the ideas presented in this thesis. To all the Robalos, my long-time friends, although they have not had a direct influence on the writing, they are people I have always been able to and continue to count on. Another thank you goes to my friends Frederico and Isabel, who always had some advice up their sleeve and always supported me, whatever my decision was. To my goddaughter Beatriz, who without a doubt is, and will always be, a driving force in my life and a reason to never give up. To my sister, not by blood, but by heart, Ana Rita, for her patience in times when I needed a good venting and for her constant support. To my girlfriend Raquel, for always being there and for motivating me and picking me up when I was down and unmotivated. Finally, and most importantly, I would like to thank my parents and my brother for everything they have done for me since day one and the trust they put in me. Sincerely, thank you all from the bottom of my heart.

This page is intentionally left blank.

Abstract

As data grows exponentially over time, it becomes a challenge for companies to cope with this growth. One of the most obvious problems caused by data growth is the increased need for data storage. Furthermore, as data grows, its complexity also increases, especially if the data is unstructured or its quality is poor, giving rise to the inclusion of volatile data. As such, the work produced for this document deals with a real problem, proposed by the telecommunications company Altice, whose problem focuses on the consolidation and improvement of data information regarding addresses. The interest and relevance of this topic are based on the problems present in the data operationalization for many companies, such as Altice, whose problems are related to data quality, such as completeness, correctness, duplications or inconsistencies. The approach presented here is designed to solve data problems, with an emphasis on the analysis of different tools and intelligent techniques to address Altice's problem, regarding data consolidation and processing. In order to allow Altice to take advantage of the functionalities developed for its problem, an architecture is proposed for building the software that incorporates said functionalities. To prove the executability and validate the approaches designed for each type of issue identified by Altice, results are presented on the approach designed concerning the correct assignment of matches between addresses, which aim to validate and prove its viability, in addition to proving its usefulness for the operationalization of Altice.

Keywords

Data Consolidation, Intelligent Systems, Information Retrieval, Databases, Record Matching

This page is intentionally left blank.

Sumário

À medida que os dados crescem exponencialmente ao longo do tempo, torna-se desafiante para as empresas lidar com este crescimento. Um dos problemas mais óbvios causados pelo crescimento dos dados é a maior necessidade de armazenamento de dados. Além disso, à medida que os dados crescem, a sua complexidade também aumenta, especialmente se os dados não estiverem estruturados ou a sua qualidade for fraca, dando origem à inclusão de dados voláteis. Como tal, o trabalho produzido para este documento trata de um problema real, proposto pela empresa de telecomunicações Altice, cujo problema se foca na consolidação e melhoria da informação de dados relativos a moradas. O interesse e relevância deste tópico baseiam-se nos problemas presentes na operacionalização dos dados para muitas empresas, tais como a Altice, cujos problemas estão relacionados com a qualidade dos dados, tais como completude, exatidão, duplicações ou inconsistências. A abordagem aqui apresentada foi concebida para resolver problemas de dados como os referidos, com ênfase na análise de diferentes ferramentas e técnicas inteligentes para tratar o problema da Altice, no que respeita à consolidação e processamento de dados. De forma a permitir à Altice tirar partido das funcionalidades desenvolvidas para o seu problema, é proposta uma arquitetura para a construção do software que incorpora as referidas funcionalidades. Para provar a exequibilidade e validar as abordagens concebidas para cada tipo de problema identificado pela Altice, são apresentados resultados sobre a abordagem concebida relativamente à correta atribuição de correspondências entre moradas, que visam validar e provar a sua viabilidade, para além de provar a sua utilidade para a operacionalização da Altice.

Palavras-Chave

Consolidação de Dados, Sistemas Inteligentes, Recuperação de Informação, Bases de Dados, Correspondência de Registos

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

API Application Programming Interface.

CTT Correios, Telégrafos e Telefone de Portugal.

ETL Extract, Transform, Load.

GPS Global Positioning System.

INE Instituto Nacional de Estatística.

NLP Natural Language Processing.

RDBMS Relational Database Management System.

URI Uniform Resource Identifier.

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Information sources nowadays are ubiquitous. However, most information sources are not human-ready, meaning that a person is often indispensable in the process of obtaining and analyzing data, as well as in the process of extracting and consolidating information.

Although this scenario is not expected to be completely overcome, the truth is that there are more and more systems and processes aimed at reducing this need, i.e., making it possible to extract information from heterogeneous and noisy sources more and more automatically (or at least automated).

At the same time as there are progressively more solutions for handling data, there are also increasingly demanding challenges with a certain degree of complexity that lead to the need to analyze, process, and act upon data from several different systems. Data analysis for consolidation and information extraction usually consists of inspecting, cleaning, transforming, and modeling data in order to discover relevant information.

The application of data analytics techniques can bring significant advantages over competitors by enabling increased efficiency, achieved by consolidating existing data, accommodating new data or removing duplicate data. For any organization that wants to communicate with its customers, members or employees, or provide comprehensive services in terms of touch points and interactions performed, the ability to recognize the same item, e.g. person, company, address, computer, on different systems (or even within the same system) is essential.

The challenges associated with information consolidation and extraction are from various sources that are not always easy to identify and handle, for example, different nomenclatures for the same thing, which sometimes gives rise to repeated data (e.g. `Nossa Senhora de Fátima` and `Nª Srª de Fátima`), data entry errors (e.g. `empreza` instead of `empresa`) or field integrity (e.g. allowing a numeric field to have strings entered).

Additionally, it is often the case that there is no global identifier to identify the same item in different data sources, which would link the equivalent records across different sources.

The challenges mentioned in the previous paragraphs are the focus of this work, which tackles a real problem and deals with the consolidation and extraction of information regarding addresses. Naturally, the interest and relevance of this subject are based on the problems present in the data operationalization of many companies. To put it in context, the topic of the work will be described in Section 1.1.

1.1 Context

In the scope of information consolidation and extraction, this work focuses on the consolidation of address information for the telecommunications company Altice. This is a very pertinent operational goal given that a large part of the services provided are based on a physical location (address).

The more specific environment will be the consolidation of Altice's main address database, which includes confirmed addresses with services provided and addresses where services may become available in the future. In this context, as this data source is not fully complete in terms of national addresses, it is important for Altice to ensure the quality and accuracy of its operational data and to absorb data from various sources to make it as complete and correct as possible. In addition to this context, there are external sources, namely from Instituto Nacional de Estatística (INE) and Correios, Telégrafos e Telefone de Portugal (CTT), which can be used to extract new information.

1.2 Objectives

The main objective of this work is the investigation and application of intelligent methods in the consolidation, extraction and enrichment of address information from heterogeneous data sources. To achieve the main goal, there are several challenges to be met, namely:

- Identification of new addresses to be added and duplicates to be removed;
- Enrichment with relevant information to the records;
- Association of location to addresses;
- Definition, implementation and integration of the solution at Altice's facilities.

Besides the main objective for consolidation purposes, this work also aims to design a system that allows the application and execution of the developed functionalities for extraction and consolidation from external sources and make possible autonomous use by Altice.

1.3 Document structure

After this introductory chapter, Chapter 2 presents the fundamentals and work related to information consolidation and extraction, namely the most widely used approaches in the context of string extraction and matching, and also references previous research on the topics of string matching for similar NLP problems and what is new in the area.

Chapter 3 presents a brief description of the data sources, as well as the requirements defined for creating the solution.

Chapter 4 presents the architecture for the solution, in addition to the defined approach.

Chapter 5 presents the implementation language and tools used, as well as the functionalities employed for each of the requirements.

Chapter 6 presents the tests and results produced for the validation of the final solution.

Finally, Chapter 7 closes the document with a summary of the results and proposals for future work.

This page is intentionally left blank.

Chapter 2

Background and State of the Art

Since data can have countless shapes and forms, many other problems related to the topic of this thesis have been researched for a long time and will continue, given the exponential growth of data with business expansion and the evolution of technology. Consequently, there are many strategies and information that can be used to address and study the problems regarding data. Therefore, this chapter is dedicated to the study and analysis of previous works on the subject and what is most commonly used in terms of algorithms, approaches and technologies that can be considered relevant for to the scope of this dissertation.

The result of this study will largely give rise to different approaches for dealing with the problem laid out in this document. For that reason, concepts like Record Linkage, Extract, Transform, Load (ETL) will be explored in this document. Furthermore, the study and application of string metric algorithms such as Levenshtein, Jaro-Winkler, N-Grams and Cosine are essential to achieve this work's goals.

2.1 Extraction, Transformation, Loading and Linkage

Data extraction and consolidation refers to the method of acquiring, processing, and cross-referencing multiple data sets and moving it to a new context. The benefits associated with consolidation are quality assurance and accuracy of information, which aids in data access, manipulation, and analysis [4]. Additionally, it can be helpful in order to improve a company's operational paradigms, for example.

The sources used for consolidation can be both heterogeneous and homogeneous, but to get into the context of our problem, the focus will only be on consolidation of heterogeneous sources. By heterogeneous sources, as stated in [37], are those that do not have uniformity among themselves, that is, with high variability of data types and formats. They are sources with different structural characteristics, and this is due to the fact that they are built on different and specific purposes within organizations.

When data is extracted from the various sources, it has to go through a cleaning process to handle redudancy, inconsistency and integrity up in order to be combined and transformed into a consumable and editable format, stored in a data repository for consultation. This process is called ETL [39]. The main goal of ETL is to prepare data for analysis or business intelligence, as it allows businesses to consolidate data from multiple sources into a single repository with data that has been properly formatted and qualified in preparation for

analysis [17][32].

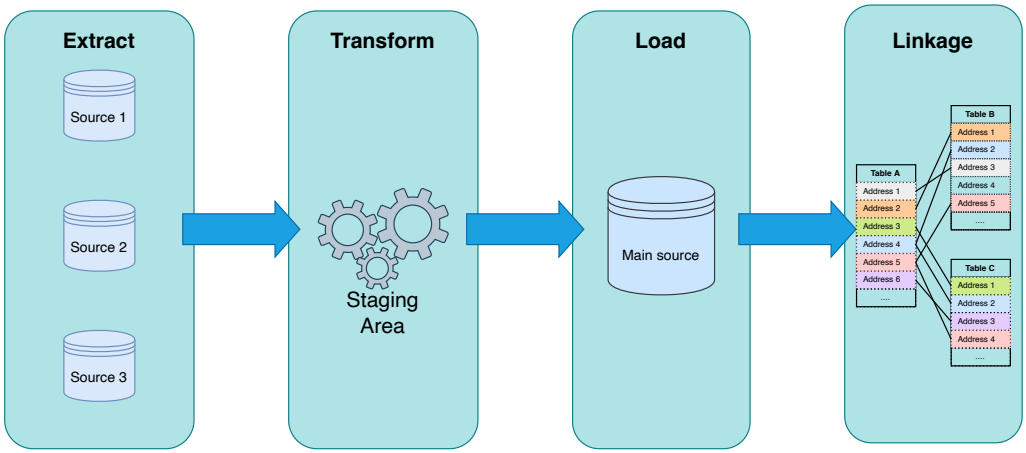

Figure 2.1: Extract, Transform, Load and Linkage process (adapted from [1])

2.1.1 Extraction

The first step is the data extraction process, which involves identifying the most relevant original sources and extracting data from them, regardless of the format, into the staging area, where all the information is converted into a single format and prepared for transformation [17]. The staging area is usually where the validation of the collected information is done, before it is transported to the target source [34].

This is one of the most important steps, if not *the* most important, as the next steps derive from the quality of the data that has been retrieved from the sources. This implies that if the data extraction is not done correctly, the data can have a certain degree of imperfections, which will compromise the quality of the data and is one of the main reasons why normally the whole process is never done directly at the target source [7]. Designing and creating an extraction process is often one of the most time-consuming tasks in ETL, because the original data can be complex and poorly documented, requiring analysis and determination of what data needs to be extracted.

2.1.2 Transformation

In a second iteration, a series of rules are applied to transform and derive the extracted data in order to fulfill the needs of an organization and the rules for the data storage solution [6]. Although this process seems relatively simple, direct mapping between columns and tables is impossible, even if the sources refer to exactly the same context, and this happens because of the different column arrangements and quantities for the same record in different sources. So it is usually necessary to perform transformation operations, typically normalizing data, removing duplicates, validate the integrity of the information collected in the previous step, etc. [32].

Table 2.1: Representation of a record with different structure rules.

Attribute	**Source 1**	**Source 2**	**Source 3**
Street Type	Rua	-	R
Street Title	Dom	-	D
Street Name	João II	Rua Dom João II	João II
Zip Code	3030-385	3030-385	3030-385
City	Coimbra	Coimbra	Coimbra

Examining Table 2.1, we can see that we are looking at data that corresponds exactly to the same, but thanks to its arrangement in the 3 sources, there is a need to clean up and deal with the presented information and structure it in the most convenient way, in order to map the record correctly [32]. To do this, normalization must be done for some types of information that do not conform to a standard format. When this format is defined, normalization will be achieved by the use of rules that satisfy the desired format.

Still using Table 2.1 as a reference, the different representations of a street, as shown, make it difficult to compare between entities. Assuming that the correct format would be that of the **Source 1**, one would have to normalize and split the street name in the **Source 2** by the "Street Type" and "Title" fields, whereas in the **Source 3** it would only be necessary to compose the abbreviations in the same fields.

2.1.3 Loading

Finally in the last step we have the loading process, which deals with storing the transformed data to its final destination, which can be a data warehouse, a delimited text file, etc. [6]. In our case, this process applies to a database. It is at this stage that decisions are made about what existing information is to be replaced or enhanced, and what new information is to be loaded, making this data business-ready to its clients [6]. In addition, it is at this stage that the frequency of data updates is usually established, which varies according to the requirements of each system [32].

2.1.4 Record Linkage

Record linkage is the process of identifying and linking records that correspond or is believed to be the same entity, within one or across multiple independent data sources, in such a way as to be treated as a single record. There are also other names referring to this practice, such as data matching, entity resolution, entity disambiguation, deduplication [20]. When records have a unique identifier, the linking process is simple, since matching is done on the basis of equality of that common unique identifier. However, since records generally lack a unique identifier across multiple sources, other common information must be compared to link the records. For example, in the case of a person, identifiers such as first and last name, date of birth, gender, residence address, etc. are used.

The first concept of record linkage was proposed by Halbert L. Dunn, in a paper entitled "Record Linkage", published in 1946. In this paper, the term is used to describe the process of assembling the most significant events in the life of a person from the day they are born until the day they die, what the author refers to as the "Book of Life" [16].

It was not until 1969 that the main theory behind record linkage was developed by Fellegi and Sunter, in their paper "A Theory for Record Linkage", in which pairs of records are classified as matches, possible matches or non-matches, using for this purpose the fields

considered common between the entities to be compared. They additionally present a formalization of the mathematical concepts of modern record linkage, where they introduced similarity functions that calculate, based on estimates, how similar two records are. Given their growing interest in applying advances in computing and automation, the mathematical concepts of this theory continue to serve as the basis for record linkage [18].

Table 2.2: Representation of a record in different sources.

Attribute	**Source 1**	**Source 2**
Street Name	Rua Cândido dos Reis	R Cândido Reis
Zip Code	3030-075	3030075
City	Coimbra	Coimbra

Using Table 2.2 as reference, in this table are attributes of a street represented in different ways in different sources, in this case, its name, zip code and location. In this concrete example, this process can be handled manually by looking at the data directly and comparing them in order to link them. Although with human intervention accuracy can be high, this practice becomes too time inefficient and error prone when dealing with a large amount of data. This is where computers come into the picture, not only to reduce the processing time required, in addition to the human factor in reviewing the data, but also for better data quality and consistency. This can be achieved using classification algorithms to help distinguish what may or may not be a match, some of which will be discussed in the following chapters.

Taking into consideration that the linkage can be computationally heavy, ideally, the data should undergo a quality assessment before record linkage is performed. As shown in the Table 2.2, although both records correspond to the same address, the Source 1 record has a different layout than the Source 2 record, which to a machine is not clear when comparing them directly.

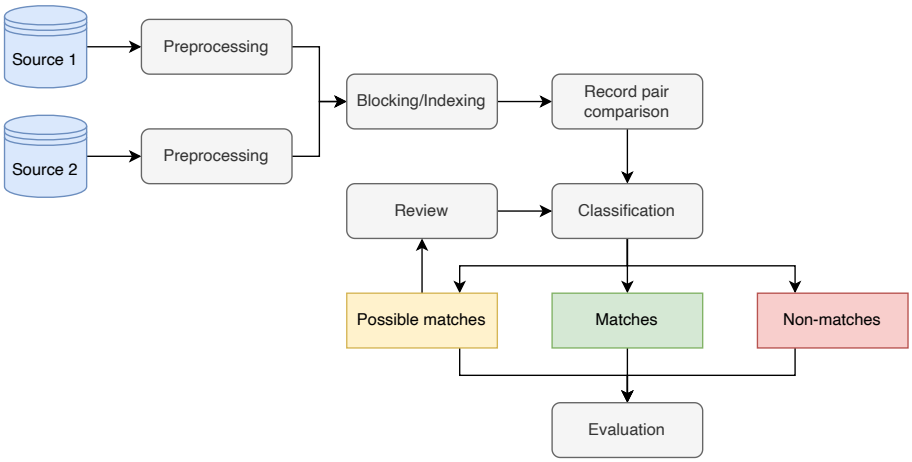

Figure 2.2: Common approach for Record Linkage (adapted from [21])

Preprocessing

For the purpose of ensuring uniformity amongst data, in this subtopic we address data preprocessing, which can be described as data cleaning as well. This approach is used on data that contains some degree of noise, such as different formatting and arrangement of

information by its fields, which for data consolidation and integration will be very costly. In order to achieve this, a transformation of the records into canonical format must be done, e.g. remove abbreviations or associated errors, normalize the arrangement of some fields. All this is of great importance, as it helps to recognize similarities between data more easily, while at the same time increasing the reliability of the equivalences [12].

Table 2.3: Representation of similar records in different sources.

Attribute	**Source 1**	**Source 2**	**Source 3**
Street Name	Rua Cândido dos Reis	R Cândido Reis	Rua Candido Reis
Zip Code	3030-075	3030075	4050151
City	Coimbra	Coimbra	Porto

For instance, in light of what is shown in Table 2.3, we can observe that identifiers for the same entity can be represented in different ways, either between different data sources or even within one's own. It is these distinct representations that worsen the application of record linkage without some sort of transformation. To obtain a higher degree of certainty and ensure that the **Source 1** and **Source 2** correspond to the same thing, ideally the abbreviation of the street name should be expanded in the second source, as well as the normalization of the zip code to have the same format. While preprocessing increases the homogeneity of the data, it further facilitates comparison between records when using similarity metrics, thus increasing confidence in the results.

Blocking/Indexing

Blocking is just as important as preprocessing, especially when we are dealing with large data sets. The larger the datasets, the more comparisons between pairs of records are required, which will cause scalability and computational cost to grow exponentially. Performing a complete scan of all records will require a complexity of $O(n^2)$.

Therefore, to avoid doing a full check on all pairs of records, the data is partitioned into groups, or blocks, so each block contains only records that are more likely to be similar [9]. Moreover, this process is intended to help minimize the number of non-matching pairs, as well as ensuring that most matching pairs are placed in their correct blocks to be linked in the following steps.

Table 2.4: Representation of records in different sources, before blocking.

Attribute	**Source 1**	**Source 2**	**Source 3**
Street Name	Rua Cândido dos Reis	Rua Cândido dos Reis	Rua Cândido dos Reis
Zip Code	3030-075	3030-075	2250-041
City	Coimbra	Coimbra	Santarém

If the attributes of the considered records are completely different, comparisons between those records are blocked. Using Table 2.4 as a reference, when looking at **Source 2** and **Source 3**, even though the street name is exactly the same in both, we realize that they are different records, judging by the zip code and different localities. By blocking these pairs, there is no longer any need to evaluate them, and the similarity techniques will only be used on the matching pairs.

Record pair comparison

From the previous step result the pairs to be compared, and from this step result the joining of the data from the sets in play. To do this, the similarity between the attributes of the pairs, usually textual strings, must be measured using appropriate comparison functions. Some of these functions will be explored in the following sections, such as the edit distance.

In our case, the similarity measures will vary depending on the type of data we are evaluating. For the case of street comparison, the goal is to find and add new information, reduce duplicates, complete missing field values, correct lexicographic differences.

Classification

The biggest challenge is at this point, when it comes to the accuracy of the results produced, since this is where the decision criteria for categorizing what will be a match and a non-match come in. A record pair can also be classified as a possible match, which is when there is doubt about whether it is a match or not.

All the data that resulted from the previous steps goes through a decision model, which will dictate what is a match from a non-match. A simple classifier to help distinguish a match from a non-match can be based on a similarity threshold, determined from the result produced in the comparison functions. Those above the threshold are merged and treated as a single record, to avoid redundancy, while those below the threshold are considered non-matches. Depending on how the classifier is being used, possible matches can rise. These cases are separated for future review, so other classifiers can be used, since there is no certainty that these pairs are the same or not [21].

Evaluation

Just as the process of record linking can add enormous value to business intelligence processes, inaccuracy in its algorithms can also be very costly. This is why it is crucial to choose a method and data attributes that will ensure maximum linkage accuracy between datasets. In this last step, the ratings are evaluated, using the true positives, true negatives, false positives and false negatives as a reference to express the comparison quality in terms of precision, sensitivity, accuracy and F_1-score [3].

- **T**rue **P**ositive - Records with at least one associated match, as supposed.
- **T**rue **N**egative - Records with no associated matches, as supposed.
- **F**alse **P**ositive - Records with at least one associated match, when they should not have any.
- **F**alse **N**egative - Records with no associated matches, when they should have at least one.

The **precision**, which is a measure of how many of the positive predictions made are correct, is given by the following equation:

$$\frac{TP}{TP + FP}$$

The **sensitivity** (or **recall**), which is a measure of how many of the positive cases are correctly predicted, is given by the following equation:

$$\frac{TP}{TP + FN}$$

The **accuracy**, which is a measure of how many predictions are correct over all predictions, is given by the following equation:

$$\frac{TP + TN}{TP + FP + TN + FN}$$

The **specificity**, which is a measure of how many negative predictions made are correct, is given by the following equation:

$$\frac{TN}{FP + TN}$$

The $\mathbf{F_1}$**-score**, which is the harmonic mean of accuracy and sensitivity, is given by the following equation:

$$\frac{2 * Accuracy * Sensitivity}{Accuracy + Sensitivity}$$

2.1.5 Summary

The previous different processes were explored to better explain how records will be extracted and handled, before entering the fields of string similarities. In Subsections 2.1.1, 2.1.2 and 2.1.3 (ETL) the steps to handle and store the data in another repository are explained. Record linkage mainly explains the process of record identification, which differentiates what is a match from what is not.

2.2 String Similarity Measures

In this section the various approximate string matching techniques that can be used for the purpose of information consolidation will be discussed and described. As there are many different algorithms and approaches for efficient searching, some of them will be explored in the next subchapters.

Approximate string matching is the technique that calculates similarity, or distance, between two strings, which is the minimum number of edit operations to transform one string into the other. The number of operations is what calculates the similarity value. Based on the obtained similarity value, we can set a threshold with the goal of determining how similar or different both patterns are, usually in situations where the number of differences is relatively small [10]. There are many different algorithms for efficient searching and, based on the properties of operations, string similarity algorithms can be classified into a set of domains, some of which are: character-based, token-based and hybrid approaches [28].

- **Character-Based** - Character-based methods, as its name suggests, treats each string as a sequence of characters. It is most suited when handling typographical errors between pairs of strings, since the basis of their transformations are minimum number of editions, between two strings **a** and **b**, needed to transform **a** into **b** [41].

- **Token-Based** - Unlike character-based, token-based methods transform complete strings (or text segments) into sets of individual words by splitting them using a delimiter. The main logic for this type of measure is to find common tokens in both sets of strings, calculating the overlap between them [41]. These methods work well when the representation rules between words, or the word order, is different (e.g, John Doe vs Doe, John), something that in character-based methods will drastically influence the degree of similarity, because the order of characters matters as an edit operation [28].
- **Hybrid Approach** - Hybrid approaches combine both ideas, in order to improve effectiveness when matching names composed of multiple tokens [41]. It extends the character-level edit operator to the token-level edit operator. For example, considering two strings like "Jane John Doe" and "Johnny Doe", two token-level edit operators can be used to transform the first one to the second one (e.g. deleting the token "Jane" and substituting "John" for "Johnny"). Like its predecessors, the token weight in the transformation are considered. So, "Jane" is less important than "John" and we can assign a lower weight for "Jane", since it is the only token that is not present in the other string [36].

2.2.1 Levenshtein Distance

Based on the search for matches, one of the most common algorithms for this purpose is the Levenshtein Distance, also known by the term Edit Distance, proposed by Russian scientist Vladimir Levenshtein in 1965, which calculates in measures the similarity between two words [27]. It has a wide range of applications within, besides matching strings, such as in spell checkers and automatic suggestions of approximate words.

The distance calculated is based on the number of **insertions**, **removes** and **substitutions** required to transform the first word into the second, which tells us that the greater the distance, the greater the difference between the words.

Typically, each operation will have a cost of 1, however, these may have other values associated with them, depending on the user's criteria. By definition, this distance can be used as a threshold to make a decision regarding the strings, whether they will match or not, in this case.

In order to define this algorithm mathematically, let's consider the descriptions for the variables in equation 2.1:

- $\boldsymbol{a}$ and $\boldsymbol{b}$ are the text strings to be compared
- $\boldsymbol{i}$ corresponds to the letter in the *i-th* position in the string $\boldsymbol{a}$
- $\boldsymbol{j}$ corresponds to the letter in the *j-th* position in the string $\boldsymbol{b}$

$$Lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{If min(i,j) = 0} \\ min = \begin{cases} Lev_{a,b}(i-1,j)+1 \\ Lev_{a,b}(i,j-1)+1 \\ Lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{Else} \end{cases} \tag{2.1}$$

Levenshtein Distance Equation (adapted from [22])

The goal of this system of equations is to evaluate whether or not they are equal, for if they are equal, no operations are required. On the other hand, if they are not equal, we increment 1 for each operation performed, which are represented by the equations contained in *min*. The first equation of this corresponds to the removal of a character, the second corresponds to insertion, and the third to substitution, so incrementation only happens if the compared characters are different ($a_i \neq b_j$).

For example, the distance between the words **'price"** and **"prizes"** will be 2, since it will take 2 operations, namely a substitution and an insertion, to reproduce the first word in the second:

1. *pri***c***e* → *pri***z***e* (replacing the letter "c" with the letter "z")
2. *prize* → *prize***s** (inserting the letter "s" at the end of the string)

This algorithm can be used to make comparisons between long strings, but it becomes impractical to reproduce the computations since the cost of the operations will be high depending on the length difference between both strings. Thus, its application becomes more feasible in topics such as record linkage when the compared strings are shorter, which also helps improve the speed of the comparisons.

2.2.2 Jaro-Winkler Distance

Jaro

Like the Levenshtein Distance, the Jaro Distance is also suitable for finding matches between sources, besides its purpose being to measure the similarity between two given words. Consequently, it is widely used in the areas of record linkage, entity linking and information extraction [38]. This similarity measure was originally proposed by Matthew A. Jaro in 1989 [23], which was developed to compare small strings, such as names. The difference between these two metrics is in the operations they perform. While Levenshtein does insert, remove and replace operations, Jaro only considers transposition operations, which in other words are the characters that occurs in both strings and match, but do not follow the correct order. Besides the transposition operations, the Jaro Distance considers the number of matching characters and the length of strings being compared [15].

In order to define this algorithm mathematically, let's consider the descriptions for the variables given by equation 2.2:

- $\boldsymbol{a}$ and $\boldsymbol{b}$ are the text strings to be compared
- $\boldsymbol{l_a}$ and $\boldsymbol{l_b}$ are the lengths of the strings $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively
- $\boldsymbol{m}$ is the number of matching characters
- $\boldsymbol{t}$ is the number of transpositions

$$Jaro(a,b) = \begin{cases} 0, & \text{If } m = 0 \\ \frac{1}{3}\left(\frac{m}{|l_a|} + \frac{m}{|l_b|} + \frac{m-t}{m}\right), & \text{Else} \end{cases} \tag{2.2}$$

Jaro's Distance Equation (adapted from [15])

The system of equations represented in 2.2 shows how the distances between the strings are calculated. For the first part, if the strings have no characters in common, then $m = 0$, then the distance between the two strings will be 0 as well, so *Jaro(a, b) = 0* and means the strings do not match. Otherwise, if the two strings are the same, then the lengths and number of matching characters are the same and no transpositions take place, so this means $m = |l_a| = |l_b|$ and $t = 0$ and consequently *Jaro(a, b) = 1* and means the strings are a perfect match. The characters are an exact match if they are identical, naturally, and these characters are not further than $\left\lfloor \frac{max(|l_a|,|l_b|)}{2} \right\rfloor - 1$ characters away [24].

To better explain how transpositions work, the words **'pahrmayc"** and **"pharmacy"** will be used as an example:

1. *p***ah***rmayc* → *p***ha***rmayc* (swapping the order of letters "a" and "h", since they are included in both strings)

2. *pharma***yc** → *pharma***cy** (swapping the order of letters "c" and "y", since they are also included in both strings)

Jaro-Winkler

In order to improve the performance in terms of similarity and to provide better results, a variant of the original Jaro Distance measure was developed by William E. Winkler and Thibaudeau in 1990, to support the idea that differences near the start of the string are more significant than differences near the end of the string, as it is believed that errors tend to occur less often at the start of the strings [40]. They are similar to a certain extent, and like its predecessor, considerations are based mainly on the order and number of common characters in the strings, but include two new variables. Therefore, the higher the similarity value for two strings, the higher the resemblance of both strings [26].

Considering Jaro's original equation (2.2), here follows the description of the variables given by the equation 2.3:

- $\boldsymbol{a}$ and $\boldsymbol{b}$ are the text strings to be compared
- $\boldsymbol{l}$ is the length of common prefix at the start of the string (up to 4 characters)
- $\boldsymbol{p}$ is the prefix scale, or the scaling factor

$$JaroWink(a, b) = Jaro(a, b) + l \times p \times (1 - Jaro(a, b)) \tag{2.3}$$

Jaro Winkler's Distance Equation (adapted from [24])

Since the Jaro's original equation (2.2) is the basis for this variant, the two new variables described above now come into play. These two variables are considered in order to increase similarity, one of which is a prefix scale $\boldsymbol{p}$, which gives more favorable scores when the first characters are the same (up to 4), whilst giving a more accurate answer to a defined prefix length $\boldsymbol{l}$ [11][13].

2.2.3 *N*-Grams

The n-grams, sometimes also called q-grams [35], are subsequences of n letters for a given word or string. These subsequences are usually divided by one, two, three, or more sequences of n letters. One-letter n-grams are called unigrams, two-letter are called bigrams, three-letter are called trigrams, and so forth [25]. For example, for the word "dissertation", with 3-character sequences the trigrams "dis", "iss", "sse", "ser", "ert", "rta", "tat", "ati", "tio" and "ion" are formed. In addition, if sequences of words are considered, the sentence "this dissertation is about database consolidation", the trigrams formed are "this dissertation is", "dissertation is about ", "is about database", "about database consolidation" [19].

The idea behind the use of n-grams is that whenever an approximate match occurs between two strings, one must resemble the original pattern. This resemblance is reflected if they share the same n-grams both in the pattern and in its approximate match [13]. Yet, another way to ensure that both strings are an approximate match is to preserve the positions of the n-grams [33].

Cosine

When it comes to modelling text, words or subsequences of strings as a vector of terms, the cosine similarity is a widely used metric due to its simplicity and effectiveness in information retrieval [30][31]. Cosine similarity is a metric that determines how similar the compared data objects are. The data objects are viewed as vectors in an inner product space, with vectors typically being non-zero, and this product is measured by calculating the cosine of the angle between two vectors and determining whether these two vectors are pointing in the same direction, regardless of their size.

The cosine similarity is described mathematically as the division between the dot product of vectors and the product of the euclidean norms or magnitude of each vector [5]. Its equation can be derived using the Euclidean dot product formula [14] which is written as:

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \, \|\mathbf{B}\| \, cos(\Theta) \tag{2.4}$$

Then, given the two vectors and the dot product, the cosine similarity is defined and represented by equation 2.5:

$$Cosine_{a,b} = cos(\Theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \times \|\mathbf{B}\|} = \frac{\sum_{i=1}^{|\sum|} a_i b_i}{\sqrt{\sum_{i=1}^{|\sum|} a_i^2} \times \sqrt{\sum_{i=1}^{|\sum|} b_i^2}} \tag{2.5}$$

In terms of vector-space approaches, the cosine similarity between representations based on character n-grams (i.e., based on sequences of n consecutive characters, typically with $n = 2$ and/or with n = 3) is a common approach [31]. However, it still cannot handle the semantic meaning of the text perfectly [30] and this occurs due to the implementation of cosine similarity measurement between two vectors sometimes yields unreliable result syntactically, when compared. Syntax matching may not be able to meet the difference of semantic meaning problem, and for this reason will be applied together with the aid of n-grams.

The purpose of using n-grams and cosine is to separate words into tokens, and then to separate each of these tokens into fixed sequences of n characters, which will be treated as vectors when using the cosine similarity. Keeping in mind that these two algorithms will not be used as main, but rather as auxiliary, due to this similarity calculation approach being relatively rudimentary. Their purpose will be mostly to confirm the veracity of the results obtained from the similarity measures, previously mentioned, and to make typographical errors detection possible.

2.2.4 Summary

In the previous, we have focused on different similarity algorithms most commonly used, studied with the intention of being combined and applied in the context of the stated problem. In addition to explaining the algorithms, the different types of string manipulation (character-based, token-based, hybrid) and operations (insert, remove, replace, and transpose) are explained to clarify how said algorithms work and how they treat transformations.

2.3 Associated Technologies

In order to accommodate the data provided, it was necessary to create the infrastructure to import and store the data received. Given the constraints of the project, it was considered to use open source software. Thus, **PostgreSQL**[1] was selected, which is a relational database system that is among the most widely used and one of the most established in the community for the SQL language, given that it offers a wide set of quality attributes, such as reliability, data integrity, extensibility, scalability, among others.

For the operationalization and development of the solutions, the **Python**[2] language was considered, both because it is one of the most widely used languages nowadays, and because of the wide range of modules available in the library and its high readability. These allow basic functionality to be included without having to write additional code, one of these being integration with PostgreSQL, with the help of the **Psycopg2**[3] adapter, which communicates directly with PostgreSQL through Python, allowing to retrieve the information needed through queries. The **FuzzyWuzzy**[4] and **Jaro-Winkler**[5] libraries, also open source, are used in addition to Psycopg2 to calculate string similarity measures. These libraries contain the Levenshtein [27] and Jaro-Winkler [40] distances to help calculate the similarity index between the compared records.

2.4 Summary

Previous work on the topic of data consolidation was explored in order to create an appropriate approach and select the processes and algorithms that best fit the problem addressed in this project and build a usable solution. Additionally, for each process and algorithm, a brief history on the development and changes over the years is presented. More specifically, processes such as Record Linkage, which deals with how records should be treated, namely

[1] `https://www.postgresql.org/`
[2] `https://www.python.org/`
[3] `https://pypi.org/project/psycopg2/`
[4] `https://pypi.org/project/fuzzywuzzy/`
[5] `https://pypi.org/project/jaro-winkler/`

the mapping between sources so that a distinction can be made between the various types of data, and how partitioning is to be handled, so it is possible to keep the solution scalable, as well as decrease the error-proneness. In addition to the processes, approximate string matching algorithms have been studied with the intention of being combined and applied in the context of the stated problem, mostly to aid in record linkage and to set a considerable threshold for distinguishing a match from a non-match.

Chapter 3

Problem Definition

Considering that Altice's main database is not complete in terms of all addresses in the national territory, the scope of this project is focused on designing and implementing consolidation and knowledge extraction processes using intelligent techniques, identifying specific and quantifiable processes to compose and improve data quality. This improvement is to be achieved essentially through four processes:

(i) Improvement of existing address information;

(ii) Identification of potential duplicate addresses;

(iii) Identification of potential new addresses;

(iv) Identification of potential addresses to be removed.

In addition to the goal of integrating information into the database, there is the intention of transforming the implemented functionalities into a service that allows its autonomous execution by Altice. This service must allow the system users to perform a series of operations on addresses, either individually to each address, or to a subset of them.

This chapter details the problem to be addressed and the main objectives to be achieved. First, the specification of the requirements is presented. Then, a description of the provided data sources, to get a better view of what the main challenge is and how it is going to be solved.

3.1 Requirements and Constraints

3.1.1 Requirements

Given Altice's needs and taking into account the nature of the problem, five functional requirements were defined that are considered fundamental to the development of this project. These requirements must be guaranteed so that the main objectives intended by Altice are met.

The functional requirements of the system are detailed in Table 3.1.

Table 3.1: Requirements for applying information retrieval techniques.

Requirement	Description
R1	Determine geographic coordinates for records in Altice's address database, based on address information from external sources.
R2	Identify for each non-certified address the corresponding certified address.
R3	Identify addresses from external sources that are not in Altice's address database.
R4	Identify addresses with different names that correspond to the same location, from external sources.
R5	Transform and aggregate the methods applied to data consolidation into an operable service.

For **R1**, it is known that in Altice's database there are addresses that have no geographic coordinates and some that come from sources that provide low confidence. Therefore, in this requirement it is intended to enrich the records with geographic coordinates present in the auxiliary information sources. The degree of certainty of the coordinates obtained must be quantified as well, so that it can be identified which ones can be loaded automatically, and which ones must be validated manually.

In **R2** two types of addresses are distinguished, certified and non-certified (tickets), with the first type corresponding to records that are accepted as official addresses, and the second referring to address records that have missing fields, which prevent its evolution, making it impossible to certify. The main purpose of this requirement is to assist in the certification of ticket type addresses and thus reduce the number of data to be manually processed and validated. Among all non-certified address records, those that already exist in the Altice database as a certified address should be identified, otherwise try to complement missing fields in order to assist in their certification evolution.

For **R3**, bearing in mind that Altice's database is not complete regarding all of the addresses that exist on national territory, in this requirement it is intended to enrich it with new addresses that may exist in the auxiliary information sources. From the addresses that are identified as possible new, a list must be created with these, which must contain all the necessary information for the creation and loading of records on the Altice side. The degree of certainty associated to the new addresses must be quantified as well, in order to identify which ones can be loaded automatically and which ones must be validated manually.

As time goes by and toponymy evolves there are streets that go extinct or change their names, which leads us to the problem related to **R4**. It is intended to identify addresses that are synonymous, that is, addresses that have distinct names for what should be considered the same, in order to improve data quality by removing duplicate entries. To do this, a list should be created with the records representing the same address, identifying the official record that should prevail.

Finally, **R5** has the objective to aggregate and transform the functionalities developed for the requirements **R1**, **R2**, **R3** and **R4** into a service that enables the integration with Altice's systems, through an API. In this sense, the engineering processes for the transformation and development of this requirement into operable software must be followed.

3.1.2 Constraints

There are a few constraints that will have an impact on the requirements contemplated, namely technical constraints. As for these constraints, it stands to notice that the technologies used must be open-source and the operating system should be Red Hat, mainly because it is the operating system used by Altice's machines. Moreover, the system developed must be able to be used by Altice autonomously. The restrictions are described below:

- **C1: Open Source** - The technologies used must be Open Source;
- **C2: Operating System** - The operating system used should be Red Hat, at Altice's request;
- **C3: Integration at Altice's facilities** - The developed solution must be installed at Altice's premises.

3.2 Data Sources

To be able to fulfill the requirements and better contextualize the problem, four sources of information are provided for the purpose of data consolidation, all located in national territory. Since the main goal is to consolidate and enrich address information, one of the sources, named **POLARIS**, will be the main target of consolidation, while the other three, named **Correios, Telégrafos e Telefone de Portugal (CTT)**, **Instituto Nacional de Estatística (INE)** and **SURVEY**, will be the data sources for adding and enhancing the information present in POLARIS, represented in Figure 3.1. In order to be able to cross-reference and compare information between sources, a few common fields are used, such as **street name**, **house number**, **floor**, **side**, **zip code** and **Global Positioning System (GPS) coordinates**.

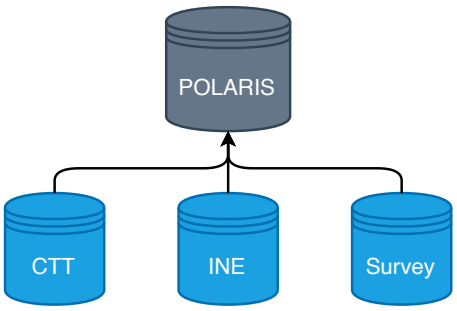

Figure 3.1: Database connection.

POLARIS is the database containing all addresses operated by Altice, which is intended to provide the addresses for all their internal systems. Within this there are two separate types of data, certified and tickets. The first type refers to Altice's official operational data, and the second refers to non-certified data, which is data that is missing information and needs complementing in order to become certified and operable. In addition, this relational database contains its address elements separated by three main types, which are streets, buildings and housing units.

CTT represents the official addresses for housing units. Although it is a consistent source in terms of quantity and quality of address information, it presents a problem concerning

the coordinates, which are often represented at the mailbox level, unlike the other sources, which generally focus on the centroid of the building. This can influence the accuracy of coordinates in POLARIS, when inherited from this source.

INE represents official addresses as well, and these are obtained and produced through the national population census study, which is carried out every 10 years. Although this source contains less information than in CTT, its geographic coordinates are way more accurate, besides being very relevant in the domain of address information. INE contains data from 2011, and for this purpose, some of the data may be outdated in terms of streets that no longer exist, which may be enough to introduce noise or unwanted addresses into the main database.

SURVEY is an auxiliary database, owned by Altice, which is based on the construction of the optical fiber footprint for customers connected to the company. This is a smaller source in terms of size, both in number of fields and total records, than the CTT and INE databases. As it refers to buildings, it may not bring much relevant information in terms of complementing housing units, for example. However, we take advantage of the accuracy of the GPS information, bearing in mind that this is the result of the study done in relation to the optical fiber operated by Altice.

3.3 Summary

In Chapter 3, the problems and objectives of this project are presented, along with the requirements. In addition, a description for each data source is provided, which gives a better insight of the problem in hands like how the databases will cross-reference each other, which information or fields should be used, etc. This chapter lays the foundations for the proposed approach and architecture that is described in the following chapter.

Chapter 4

Approach and Architecture

This chapter details the approach designed to address the proposed challenges, as well as the software architecture defined for building the solution. First, an introduction explaining the need for a customized approach is presented. Then, the customized approach is put forward, with a thorough analysis of the steps and methodologies applied to each iteration for information consolidation and extraction. Finally, in combination with the defined approach, an architecture is also presented, first to contextualize the design and then a more conceptual one using the C4 model [8] that facilitates understanding by the intended audience for each level.

4.1 Approach

The designed approach needs to meet the requirements described in the last chapter (see Section 3.1), defined to achieve four different goals regarding information extraction and consolidation, notably, improve existing information; remove duplicates; add new information; and remove wrong information. This fourfold process makes it necessary to design a specific approach, considering that no "silver bullet" solutions or ready-to-use tools were found in the literature.

Hence, the approach was designed to take advantage of several techniques that complement each other. The exploration of these techniques and the understanding of how they interact with each other was done in order to achieve the best possible results, given the fact that it should be possible to validate the integrity of the data.

The data processing practices are based on what was explored in Chapter 2 and what is considered state-of-the-art, namely, information retrieval, record linkage, and string similarity measures.

Given the size of the data and the techniques used to compose the project, the approach to the problem must be scalable to the point of avoiding that execution times or required resources grow faster than the volume of records to be treated. In order to achieve sub-linear complexities, the solution must consider an effective subdivision and partitioning of the data, coupled with indexing strategies, derivation of that data and materialized views.

Figure 4.1 is an overview of the suggested approach. Therefore, this will serve as a reference to explain the steps and techniques used in each.

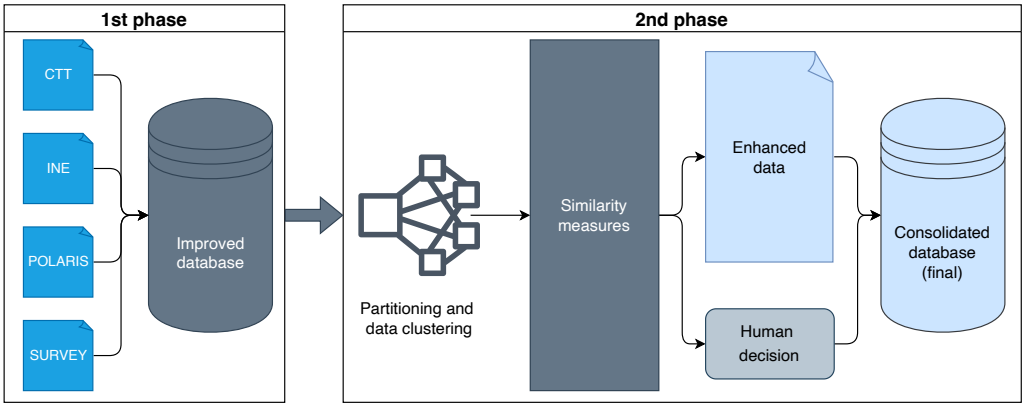

Figure 4.1: Overview of the proposed approach.

In the first phase represented in Figure 4.1, the goal and main focus is to apply information retrieval techniques to the different databases provided, in order to validate, integrate, index and improve them in an external environment. To better explain the process, a more detailed overview of the procedures will be presented in Subsection 4.1.1.

The second phase focuses on processing and enhancing the data, and for this purpose clustering analysis techniques will be addressed, followed by string similarity measuring techniques, based on Natural Language Processing (NLP). The procedures in question will be explained in more detail in Subsection 4.1.2.

It is important to point out that the defined approach may change and is in continuous improvement until the end of the project, and that it is natural that some steps are more undefined, meaning that although there is an idea of the high-level path, it will still be necessary to detail and optimize the steps according to the performance of the various techniques that are being used.

4.1.1 Data Integration and Enhancement

This subsection discusses the first phase of the Figure 4.1 activities, which aims to integrate the sources provided in an external environment, with the purpose of creating an improved database, in order to make its processing as scalable as possible. To this end, all sources must be validated, so that it is possible to relate the data to each other and efficiently analyze the various types of cases that we will encounter and design the appropriate strategies for each of them. This requires several steps, including data integrity validation, data integration into a relational database, attribute mapping, partitioning, indexing, and feature extraction. Figure 4.2 provides a detailed overview of this step.

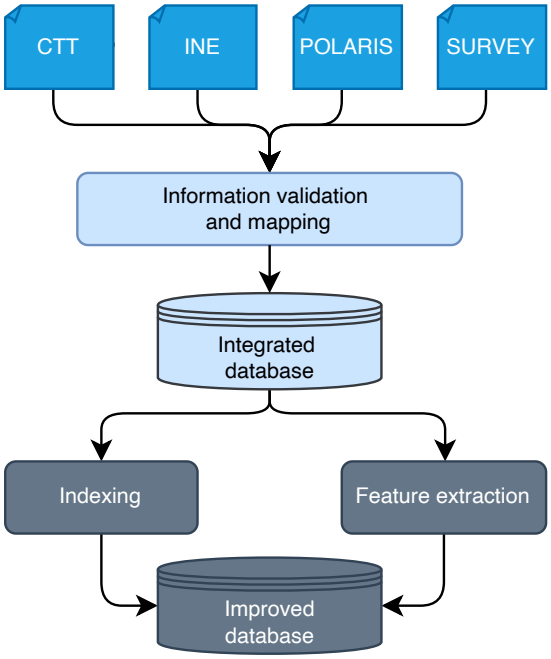

Figure 4.2: Detailed view of the the data integration and enhancement phase of the approach.

Firstly, it is necessary to **validate** the integrity of each of the data sources individually, in order to correctly map all the information and analyze the relationship between the different columns. To this end, efforts were made to understand the structure of these data sources with the help of associated documentation, besides counting on support and information provided by Altice as well. With the knowledge absorbed from these efforts, data integrity validation is done so that the data can be loaded into a relational database engine, in order to make the data manipulation efficient and scalable in the next steps.

After the validation phase, the correspondence between fields or sets of fields in the various tables provided is mapped in order to understand how the various databases can be related to each other. Therefore, the first **integrated database** contemplating all the supplied data and respective relations is obtained.

Finally, it is necessary to take advantage of **indexing** and **feature extraction**. Indexing is intended to improve the performance of the database when it comes to searching. Each index allows the server to retrieve specific rows, where it is located. These can be created using one, multiple columns or using partial data. Feature extraction processes allow pre-computing values derived from the remaining values, in order to simplify or speed up subsequent steps or even make it easier for humans to understand the data. After these steps are achieved, we reach what is refered to as an **improved database**.

4.1.2 Clustering and String Similarity Measures

The second phase of activities aims primarily at analyzing the data in the new and improved database, with a view on identifying data that can be improved or completed. Figure 4.3 presents an overview of the methodology.

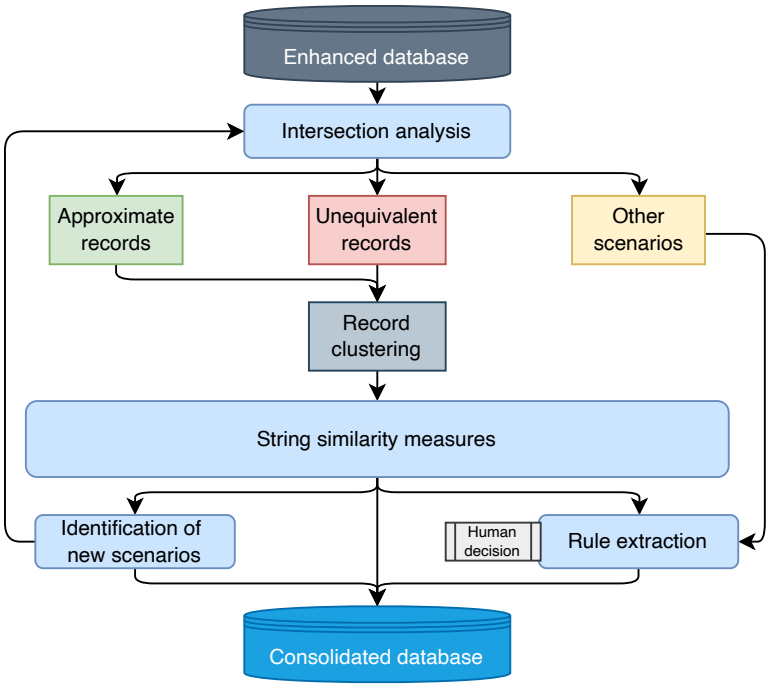

Figure 4.3: Detailed overview of the clustering and use of string similarity measures phase of the approach.

The first step is the **intersection analysis** of the different databases based on the various data in common, such as coordinates and/or zip codes. This analysis allows two types of data groups to be identified: **clusters of records that are close geographically** and groups of **records that have no counterpart** in the other databases, in order to gain some sensitivity about what can be expected from the results of the next steps. It may also allow groups of records that correspond to **other scenarios** that are not yet identified.

Given the large volume of records, in order to prevent the time complexity of the solution from being exponential, this data must be **partitioned and grouped** by reference values, in order to prevent the comparison of the data by brute force. Again, since Altice's database is not fully complete, there are cases where the data used as a reference for the partitioning is not objective or not totally correct. These cases are separated from the rest and are left for later analysis.

Based on these groups, **string similarity measuring techniques** are applied with the goal of understanding, within each group, which pairs have higher degrees of similarity. Several techniques are used for this, one of them being based on Levenshtein Distance [27], explored in Chapter 2, this being one of the focuses in view of using similarity measurement techniques. However, to robust the veracity of the results, other techniques, like Jaro-Winkler, n-grams and cosine similarity, will be used as well.

The results obtained by the string similarity measuring techniques will allow the **identification of new scenarios of interest** and also **new rules** for the records and possible different approaches for each of the requirements. In some of these cases **user validation** may be required, but one of the main goals of this project is that this will always be done on a very small number of records, in relation to the amount of data handled.

Since the topic is about residential addresses, some examples of objective and effective partitioning are grouping the first 4 digits of the zip code or by the complete zip code (7 digits). The similarity measurement techniques are applied to local clusters in order to

refine these clusters and consolidate the information for each address. In Subsection 4.1.3, the strategy of how to incorporate these string similarity measurement techniques, as well as the selection criteria for these are presented.

4.1.3 Consolidation Strategy

In order to put the previous sections into practical terms, the strategy applied to the data processing will be explained here in more detail.

Since the data originates from completely different environments than the one being developed, the process starts by checking the integrity of the data. This validation is done so that all data can be successfully integrated, since sometimes, some records conflict with the used relational database for containing characters that are not allowed. Some examples of these conflicts are broken lines (containing new line characters, splitting the record into two or more lines) due to exportation from the original source, unclosed quotation marks, which in our environment is interpreted as an unterminated string, fields that contain the delimiter inserted in the column text, which the system understands as an extra column.

Once the validation phase is over, the data is loaded into the system, where all the sources go through a normalization process on the fields that are common. Normalization must be done since we are dealing with distinct sources, which means that the rules and data layouts will also be different amid themselves. Among the common fields, one of the main fields for comparison is the complete street name, which is usually divided by street type, street title and street name.

The street type, as its name suggests, represents the type of the street (e.g. `Avenida`, which translates to `Avenue` in English) and is placed before the street's name. The street title, which is a supplementary element to the address when it is named after someone (e.g. `Rua Doutor Salgueiro Maia`), is placed between the type and the street name. Since each data source has different ways of its representing information, now and again the street type and title contains abbreviations (e.g. `AV` means `Avenida` or `DR` means `Doutor`). Naturally, different representations for the same thing will affect the accuracy of the similarity measures, hence the need to normalize the sources. For a better understanding of how many abbreviations exist in the different sources, refer to Appendix A.

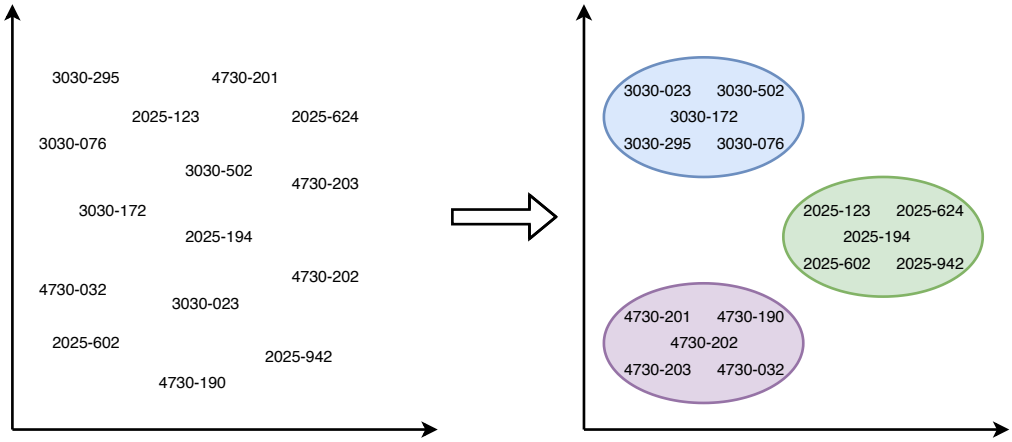

Figure 4.4: Example of a grouping by zip code.

Now that the data is more uniform, it is grouped by its respective zip codes, as shown in

Figure 4.4, to maintain acceptable scalability when the similarity measurement is being performed. In order to include only the relevant information from each source and make the comparison process modular, the identifier, address and door number of each source are included.

Next, the data grouped in the previous step is compared using similarity measures. To do this, the addresses are compared using textual similarity and only those with a high degree of similarity are accepted, e.g. those that are certain to be exactly the same. At the same time, the door numbers are compared using equality, i.e. they are either the same or they are not. In this process more than one correct match may be found for the same record, and this happens for two reasons: when a record exists in multiple sources (e.g. one-to-one) or when we have repeated data in one or more auxiliary sources (e.g. one-to-many). These cases are handled in post-processing, in order to filter and select one of the matches, if it is possible to determine the correct one.

In post-processing a priority between sources is set, determined based on the degree of confidence one has in the information from each source. This step aims to break the tie between cases that contain more than one equivalence defined by the similarity measures. To better describe the priority, it goes something like this:

- If we find **one** match in the first source and **five** in the subsequent ones, since the first match is unique and is our priority, that match is chosen as correct.
- In another case, if the first source does not have any matches, the priority shifts to the second source, and if there is a single match in it, it is considered as correct.
- In the last case, if **two** matches are found in the first source, those two and the rest of the matches found in the other sources are separated, for the reason that if the source we are most confident of has repetitions, we cannot be sure which entry is correct, even in the remaining ones.

Once the priority is defined, all the data is grouped and sorted by its identifier and by the source in which the matches were found. Finally, an iteration is performed on the sorted data, and based on the defined priority, the records with unique matches are selected and those that do not respect the priority rules are separated.

4.2 Architecture

In this section, the description and proposition of the project's architecture will be detailed, based on what is wanted in terms of operational services by Altice. Furthermore, since the main goal of this project is to adapt what has been developed for Altice, the proposal for the architecture will be made with the intention of facilitating integration into Altice's ecosystem for its own benefit. The architecture to be introduced results from the requirements described in Chapter 3, mainly the last requirement (R5), which in short is the transformation of the first four requirements (R1, R2, R3 and R4) into an operable service, proposed by Altice.

Before moving on to the main subject, which is to describe and document the software architecture, a simplified concept of the system will first be presented. Then, to hierarchically document the architecture and explain how the system will be modeled, the C4 model will be used.

4.2.1 Simplified Concept

Since Altice's main objective is to produce a data consolidation system to aggregate information to its main database, the diagram represented in Figure 4.5 demonstrates the basic concept of what this system will be.

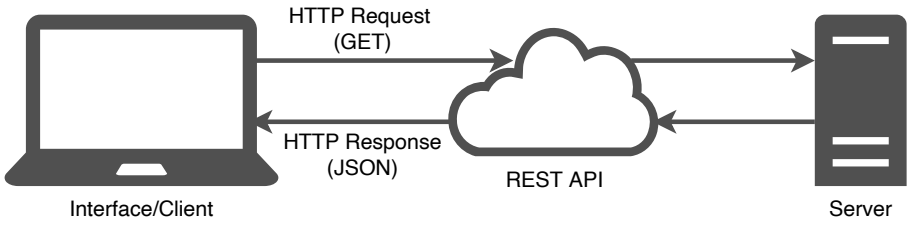

Figure 4.5: Simplified architecture concept of the service to be developed.

On one end we have the interface, which will make the requests and handle the inputs for the requests to be made. On the other end, we have the server, which will contain all the data and functionalities designed for the feasibility of what is proposed in the requirements. In the middle, we have an Application Programming Interface (API), which will serve as the communication link between the interface and the server. APIs offer security by design because their position as an intermediary eases the abstraction of functionality between two systems. The API endpoints also decouples the consuming application from the infrastructure that provides the service.

To briefly explain how communication works between the three parts, the interface initiates an API call to make requests to the server. These requests are processed from an interface to the server via the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body. After receiving a valid request, the API makes a call to the server containing the data, which in turn sends a response to the API with the requested information. The API transfers and presents the data to the initial requesting interface.

4.2.2 C4 Model Architecture

Context

Similar to what is represented in Figure 4.5, we have Figure 4.6, which corresponds to the first level of the C4 model. This diagram provides a good starting point for someone unfamiliar with the system, showing how the system interacts without details about how it works. It also shows how the software system in scope fits into the environment and its relationship with users and other systems.

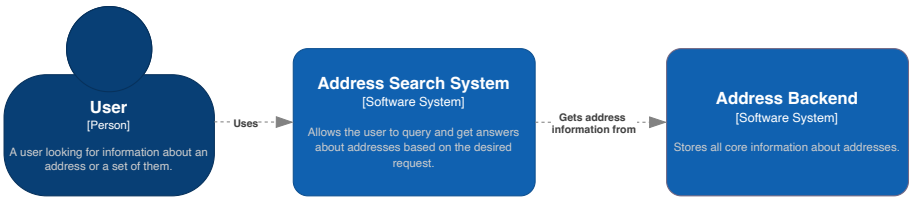

Figure 4.6: System Context diagram.

This context diagram helps us understand that two systems are needed to be able to fulfill the purpose of address consolidation. An address search system that displays a set of addresses based on the request made by the user. Secondly, a storage system is needed to manage the data and send it to the address lookup system for processing.

Container

The container diagram represents the individual service or application, and breaks the system box into "containers" that represent the execution code or store data, such as applications, databases and file systems. To fully cope with the challenges that arise from data consolidation, different technologies are required. These technologies are depicted at a high-level in Figure 4.7.

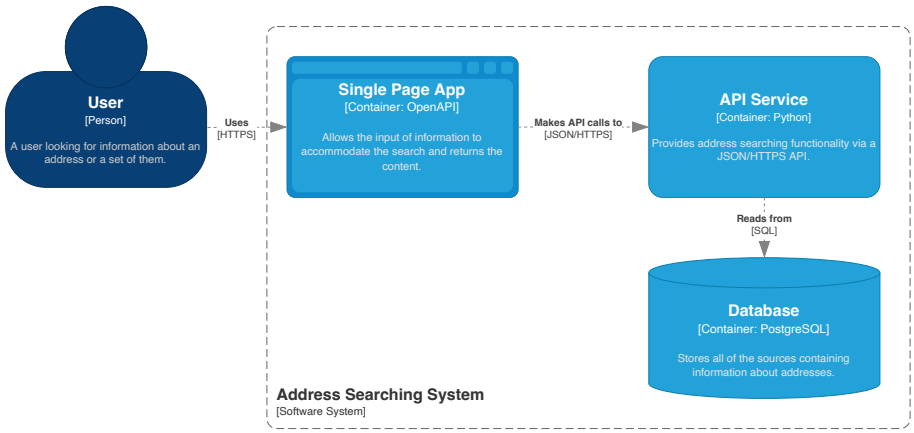

Figure 4.7: Container diagram.

It is represented in Figure 4.7 that the address lookup system is composed of three containers: a single page application, an API Service, and a database. The single page application uses JSON/HTTPS API, which is powered by an API Service application using Python. The API Service reads information from a Database, using SQL queries. The Database uses a Relational Database Management System (RDBMS) named PostgreSQL to store and read the data.

Components

The component diagram further shows how individual containers are made up of a number of "components", what each of those components are, their responsibilities and the technology details. In-depth details about the API Service container are provided in Figure 4.8.

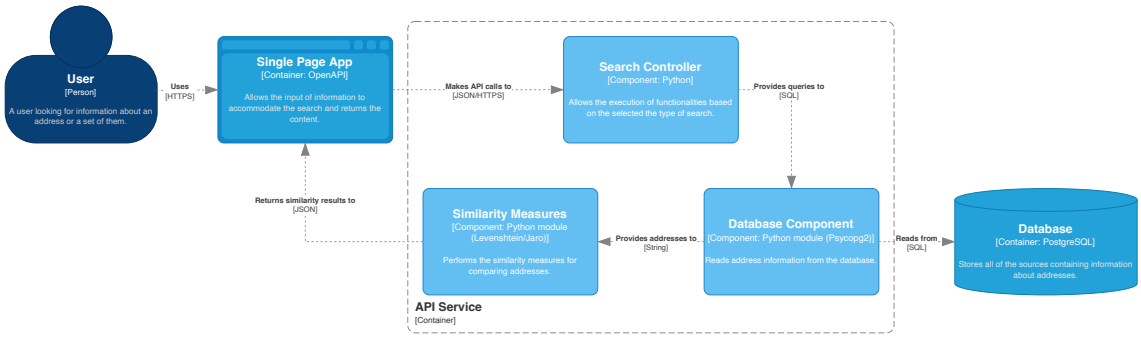

Figure 4.8: Components diagram.

It is represented in Figure 4.8 that the API Service is composed by three components: a Search Controller, a Database Component and Similarity Measures. The Search Controller uses Python to translate the requests made by the user and prepare them for communication with the database. The Database Component uses a Python module called Psycopg2, which allows communication of requests in SQL format between Python and PostgreSQL. Finally, the Similarity Measures component also uses Python modules, namely Levenshtein and Jaro-Winkler. The similarity measures included in this component compare the addresses obtained from the database and return an answer.

4.3 Summary

This chapter detailed the designed approach, as well as the software architecture defined for building the solution. The next chapter will delve into the development process details, which will implement the different systems needed to enable communication and the execution of the solution.

Chapter 5

Implementation

In this chapter, the solution development will be presented. The implementation of the solution is based on the architecture defined in Section 4.2, as well as the technologies chosen for this purpose. First, the implementation language and tools are presented. Next, the functionalities, along with an explanation of the procedures for each requirement, are detailed.

5.1 Implementation Language and Tools

Bearing in mind that one of the constraints of the project (and the most impacting on the decision about which language or tools to use) is to use only open source technologies, The Python language was selected, considering that it is an object-oriented language with a wide range of libraries, besides being one of the most popular programming languages in the world. Along with the mentioned advantages, the ease of learning and readability is also a determining factor in the event that someone else has the need to modify the code of the developed solution.

5.1.1 FuzzyWuzzy

FuzzyWuzzy is a free and open source Python library that has been developed by SeatGeek [2], which is used for the purpose of string matching, given a pattern. It uses Levenshtein Distance to calculate the differences between sequences. As discussed in their blog, the original use case was based on finding out whether two event ticket lists refer to the same event in real life. Although our problem is not related to tickets, the use case applies to our problem, since the objective is to verify what is a correct match and whether it refers to the same address. For these reasons, it was chosen as one of the implementations to support the comparison of the addresses.

Within the FuzzyWuzzy library, some variants of the Levenshtein Distance are provided. Firstly, we have Simple Ratio, which corresponds to the base implementation of the Levenshtein Distance. Then, Partial Ratio is also a variant which offers higher scores over Simple Ratio, taking into account that the size of each of the strings is not relevant, as long as there are at least substrings that are the same. Next is Token Sort Ratio, which gives higher scores if the substrings are contained within the strings, but have their order changed. Finally, we have the Token Set Ratio, which is similar to Token Sort Ratio, but a bit more flexible. In this last variant the two strings are tokenized, but instead of

sorting and comparing immediately, the tokens are divided into two sets: intersection and remainder. After that, the divided sets are used to build a comparison string. As in our case the strings usually always have a defined order and similar lengths, the Token Set will be used, given its flexibility.

5.1.2 Jaro-Winkler

Like the FuzzyWuzzy library, it is a free and open source Python library that was developed by Richard Milne [29]. It is also intended to be used for string matching purposes, given a pattern. The difference is that this technique will be used as an extra validation step in addition to Levenshtein Distance, since its operations for comparison are different from the latter. The choice was made having in mind a greater assurance about the veracity of the results.

5.1.3 *N*-Grams and Cosine

In addition to the Jaro-Winkler Distance helping to confirm the results of the Levenshtein Distance, the n-Grams and Cosine techniques are also used, with the goal of making possible the detection of typographical errors. While n-Grams separates words into tokens, and then separate each of these tokens into fixed sequences of n characters (in our case being being tri-grams). In turn, these sequences are transformed into vectors, so that it is possible to calculate the distance between them using Cosine similarity.

Since the validation is already done by Levenshtein and Jaro-Winkler Distance, the probability of it being a matching error is very low. Thus, the goal is to check whether some of the sequences have a lower degree of similarity, which raises the hypothesis that this sequence is connoted as a typo.

5.2 Functionalities

This section aims to lay out the designed functionalities, explaining in detail the steps that are necessary to achieve what is intended in each of the requirements. Even though all the requirements have different goals, they all have common parts, namely the way data is clustered and compared. To ease and standardize some of the steps, the creation of address classes was implemented, where all the common fields to the sources are included, some of them being street name, number, floor and side of the door, zip code, longitude and latitude, etc.

5.2.1 Functionalities for R1

Since there was a need to have a starting point, this was the first requirement to be addressed, given that compared to the other requirements, this was the simplest to address in terms of procedures. The decision to address this requirement first was to help gain some sensitivity about the procedures involved as well.

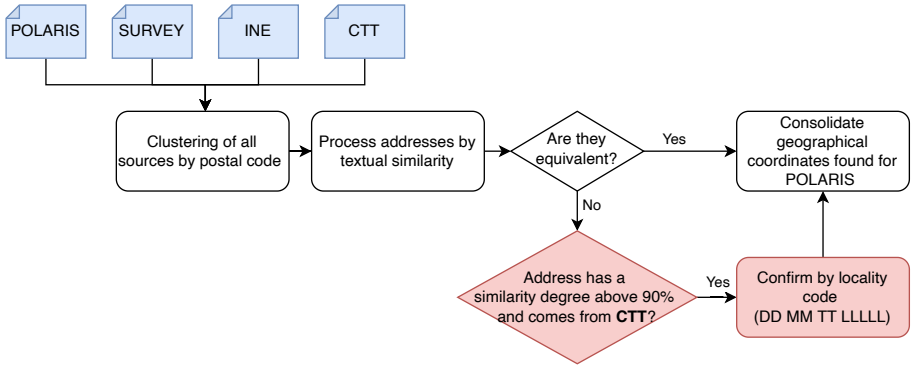

Figure 5.1: R1 flow diagram.

Using the flowchart shown in Figure 5.1, the process for this requirement starts with grouping the source data by their respective zip codes. Then, the data is compared using the similarity techniques implemented. The data is always cross-referenced between POLARIS and the other sources, always toward the consolidation of information for POLARIS. In turn, the similarity scores are evaluated between 0 and 100, with 100 being the highest score possible and the one that presents us with the greatest certainty of the equivalences. In case we find equivalences with a score higher than 90, which can lead to a false positive, further validation is done with the source CTT, which has a common locality code with POLARIS, serving as a tiebreaker. If the locality code matches between POLARIS and CTT, this match is accepted as correct and added to the set that was validated in the previous steps. Finally, after all equivalencies are properly validated, the geographic coordinate information is replaced or added to POLARIS, transforming the data into an output format defined by Altice. For a better understanding of the output formats, refer to Appendix A.

5.2.2 Functionalities for R2

After implementing the solution for the first requirement and proving that the used techniques worked, we moved on to the second requirement. This requirement was tackled second because it added another layer of complexity, taking into account that it dealt with records that were not fully validated, i.e. had missing information, which in turn caused some steps to be added to what was developed for the first requirement. Unlike the first requirement, the procedures in this requirement assist in the certification of addresses, i.e. POLARIS tickets. The SURVEY source is not used, since it is a source with few fields compared to POLARIS, making it impractical to use for information consolidation purposes. In exchange, the main POLARIS source is used as a consultation source, given the affinity between the former and POLARIS tickets.

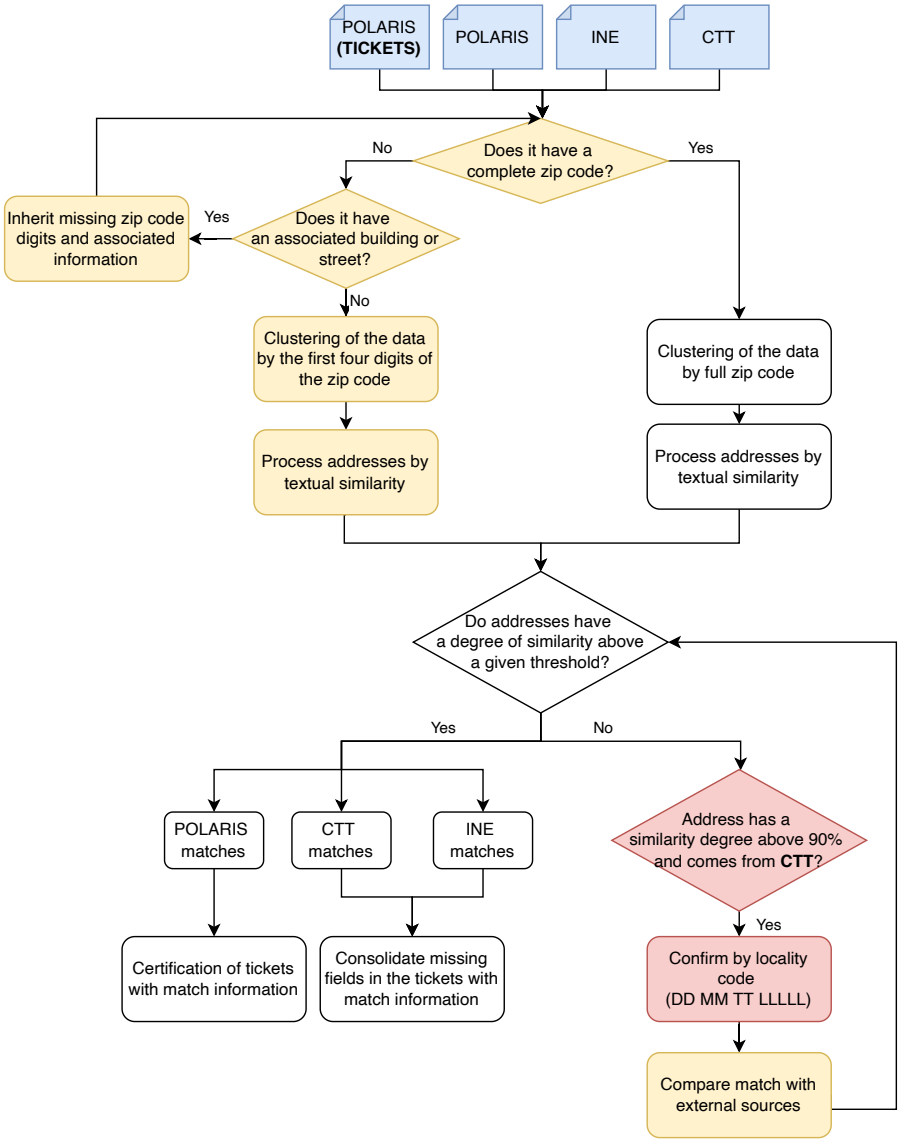

Figure 5.2: R2 flow diagram.

Using the flowchart shown in Figure 5.2, the process for this requirement begins, similarly to the previous one, with grouping the source data by their respective zip codes, if the information in question has the full zip code. Otherwise, the data is grouped by partial (first four digits) zip code, so as not to mix relatively good data with data that has poor quality. In partial grouping, the data is compared with the goal of at least finding the last three digits of the zip code first. Then, all data is compared using the similarity techniques, given a threshold. Same as before, in case we find equivalences with a score above 90, a posterior validation is made with the CTT source. The process differs from this step in relation to its predecessor, since if we find a match with POLARIS certified, the ticket in question can be automatically certified as long as the certainty about the match is very high. If a match is found with the CTT and INE, to leave no margin for error, the tickets are filled in with the information available from these two sources and transformed into the appropriate output format.

5.2.3 Functionalities for R3

In this requirement it is intended to enrich it with new addresses that may exist in the auxiliary information sources, namely CTT and INE. As with the second requirement, the SURVEY source is not used, since it is a source with few fields compared to POLARIS, making it impractical to use for information consolidation purposes.

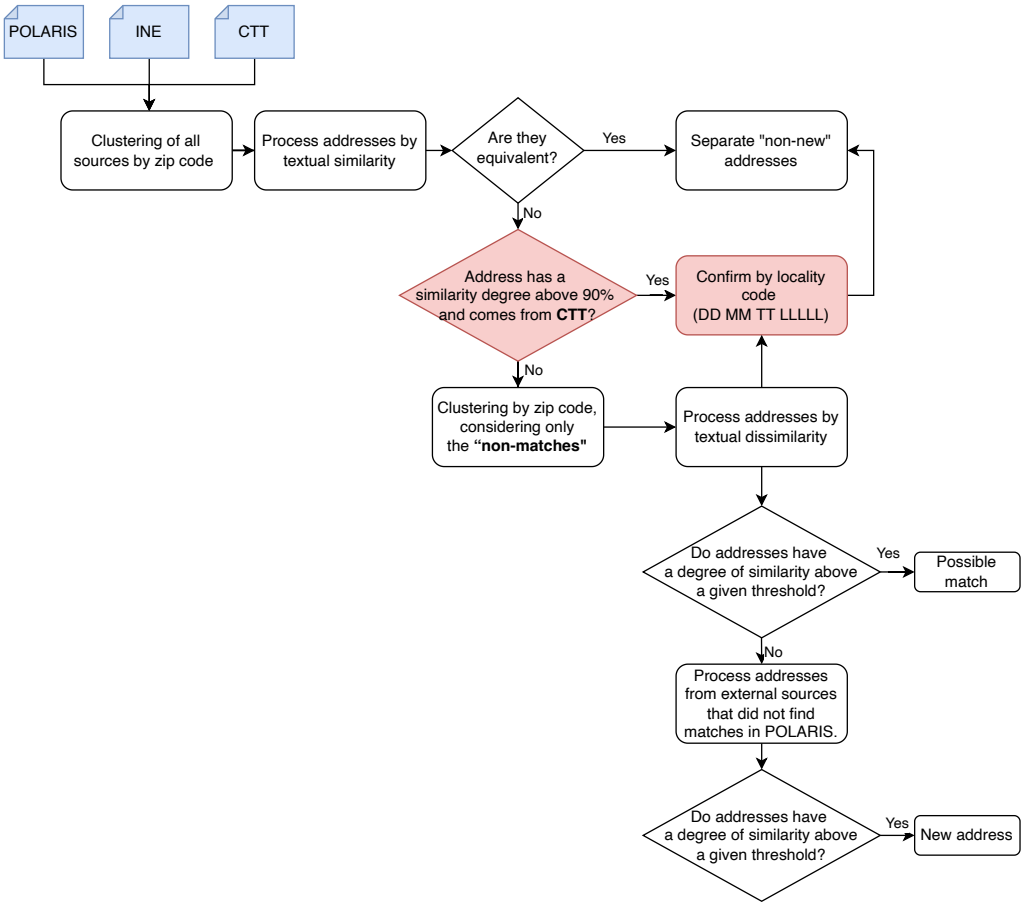

Figure 5.3: R3 flow diagram.

Using the flowchart shown in Figure 5.3, the process starts with grouping all the source data by their respective zip codes. Then, the data is compared using the similarity techniques and the data that is deemed to be equivalent is separated from the data that has no matches in the following steps. The "non-matches" are again grouped by zip code and compared towards CTT and INE, in order to prove that they really do not have close matches, which we refer to as dissimilarity. For the data to be treated as different, the similarity score must be very low. After this stage, an extra step is taken in order to check if some of the addresses given as dissimilar exist only in CTT and INE, comparing the former and the latter, respectively. If so, for the addresses identified as possible new ones, new tickets are created and then validated by Altice's systems, provided that they respect the POLARIS entry rules and have sufficient information to justify their entry in Altice's source.

5.2.4 Functionalities for R4

This requirement is intended to handle synonymous streets, that is, addresses that have distinct names for what should be considered the same. Considering that similarity techniques cannot directly validate synonymous words, we resorted to the CTT database which contains an external table with some of the synonymous addresses. As such, the comparison is made only between POLARIS and CTT.

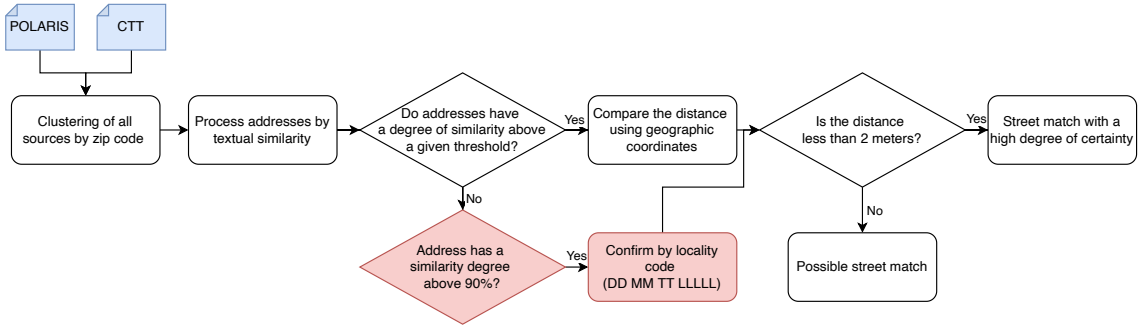

Figure 5.4: R4 flow diagram.

Using the flowchart shown in Figure 5.4, the process for this requirement is very similar to the first requirement, taking into account that the grouping and comparison is done with information from POLARIS and CTT. The synonymous CTT addresses point to the one that is considered official in their database, and as soon as there is a direct match between the official CTT address and the POLARIS address, the associated CTT synonymous addresses are consolidated into POLARIS. The biggest difference from the first requirement is the final step, which consists in calculating the distance as a way of validating whether the addresses being compared are close to each other. Like its predecessors, the list of records obtained is transformed into the output format proposed by Altice, which aims to input the streets that are considered synonymous, identifying the official record that should prevail.

5.2.5 API Endpoints

An endpoint is a remote computing device that communicates back and forth with a network to which it is connected, in other words, an entry point typically available in REST APIs. Since for each requirement a specific operation is desired, four endpoints were defined in order to realize the four operational requirements (R1, R2, R3 and R4). Furthermore, since this data will be treated in an environment external to Altice's main systems, the defined endpoints only do **GET** operations, since the data will only be for consultation and later consumed by Altice's internal systems. The list of endpoints is as follows:

- **GET /rf1/{geo_filter}** - This endpoint is used to look for missing geographic coordinates or change those that are dubious, within a chosen set.
- **GET /rf2/{geo_filter}** - This endpoint is used to certify, or help in the evolution of certification of tickets under validation, within a chosen set.
- **GET /rf3/{geo_filter}** - This endpoint is used to search for addresses present in

external sources that do not exist in POLARIS, in order to consolidate new information, within a chosen set.

- **GET /rf4/{geo_filter}** - This endpoint is used to identify addresses that are synonymous, i.e., addresses with different nomenclatures but corresponding to the same location, within a chosen set.

In addition to the endpoints, it was also necessary to use geographic filters (geo_filter fields in the endpoints) as input data in order to replicate the clustering explained in the previous chapters, defined based on the fields present in POLARIS, in order to be able to focus only on specific subsets, chosen at the user's discretion. The list of geographics filters is as follows:

1. **All zip codes** - Information search considering the entire POLARIS universe, clustered by all zip codes.
2. **Zip code** - Information search considering a specific zip code (e.g., 3030-076).
3. **Partial zip code (CP4)** - Information search considering only the first four digits of the zip code (e.g., 3030).
4. **Locality code** - Set of codes present in POLARIS and CTT used to divide the search by a specific District, Municipality or Town, in addition to the street Locality code (e.g., DDMMTTLLLLL).
5. **Primary Identifier** - POLARIS unique identifier field used to identify a specific record.
6. **Georeferencing source** - Set of codes that indicates the system that provides the geographic coordinates of the records present in POLARIS.
7. **Timestamp** - POLARIS field that indicates the date of record entry/modification on a given date, sorted by year, month, day, hours, minutes and seconds (e.g., YYYYMMDDHHMMSS).

5.3 Summary

This chapter detailed the implementation language and tools used, along with the functionalities of the solution, as well as a step-by-step walkthrough of each requirement and the API endpoints. The next chapter will elaborate on the details of the validation process of the techniques used, as well as the results obtained, which will prove the feasibility of the implemented solution.

Chapter 6

Validation and Results

This chapter details the testing environment, scenarios and results, in order to detail the usefulness of the chosen tools in the context of the project. It will start by showing the testing environment and explain the the machine's specifications. Next, the different testing scenarios chosen, defining each one and, in the end, the result of each test scenario and achieved gains.

6.1 Testing Environment

In order to validate the developed solution, a test environment needed to be developed. For safety reasons, the machine was set up on the DEI's premises with the following specifications:

- **OS** - Ubuntu 20.04.5 LTS (Focal Fossa)
- **Disk** - 800 GB
- **RAM** - 32 GB
- **CPU** - Intel® Xeon® Processor E5-2650 v4 @ 2.20GHz - 16 vCPUs

From the beginning, for the executability of what is proposed, Altice periodically provides a collection of files with the information regarding the data sources in CSV format. These repositories include both Altice's main database, named POLARIS and the auxiliary sources CTT, INE and SURVEY. As all these sources are constantly evolving, it is of interest to maintain regular updates on the data in order to include new information, either for treatment or for reference. In addition, these refreshes also deal with the removal of discontinued data. Table 6.1 presents the size in numbers for each data source used in our experiments.

Table 6.1: Summary of the sources provided.

Data Source	Total	# Buildings	# Housing Units	Observations
POLARIS (certified)	13 396 671	5 182 254	8 214 417	Only certified addresses
POLARIS (tickets)	13 587 648	1 045 387	12 542 261	Addresses in validation process (not certified)
SURVEY	3 241 799	3 241 799	-	Only contains building addresses
CTT	6 905 177	4 509 294	6 905 177	Buildings identified from the housing units
INE	5 910 282	3 566 268	5 910 282	Buildings identified from the housing units

In Table 6.1 the data is separated by their type, in this case housing units and buildings. In POLARIS a distinction is made between buildings and dwellings, unlike CTT and INE, which only have information on housing units, and buildings are obtained from these units. This distinction is necessary to be able to separate the data according to the context to which the data is being applied. A perfect example of why this separation is necessary is, for example, when we are looking for geographic coordinates, it doesn't make much sense to include this information in the housing units, since these units are contained in the building, making it useless to have this information repeated by the different housing units. Another reason is that the SURVEY data source only contains data about buildings, which also makes it necessary to separate the data types.

6.2 Testing Scenarios

The test scenarios will revolve around the precision of the similarity measures in getting correct matches for each requirement. Then, to evaluate the performance of the similarity measures in obtaining correct matches on the different requirements, different thresholds on the similarity measures will be tried in order to evaluate which tuning is best to avoid false positives and aggregate as many true positives as possible. Furthermore, to avoid matching errors (streets that have the same name but are in different locations, for example) and to prevent unnecessary comparisons, all data is always grouped by their full zip code (7 digits), but within their partial zip code (4 digits). In order to not consider the entire universe of data, but still have an acceptable amount of data, for each requirement a subset is chosen in a partial zip code, which varies as needed for the requirement in question. This decision is also due to the evaluation of the amount of true positives vsfalse positives obtained in the similarity measures being made by looking directly at the data in order to validate the veracity of the results obtained. For example, the first requirement is to include geographic coordinates for records in POLARIS that do not have them, the zip code was chosen based on the largest number of records without coordinates.

To recap in part how the similarity measures are implemented, the thresholds of all similarity measures vary on a scale from 0 to 100, which is interpreted as the percentage of similarity between matches in our context. So, in each zip code (7 digits) the comparisons are processed using the Levenshtein Distance, and only those with a 100% score are accepted, since below this score there is the possibility of gathering some false positives. The addresses with a score below 100 go through a revalidation where they are compared using Jaro-Winkler and n-Grams. The rationale for this revalidation is to ensure that we are not losing good matches due to errors (spelling, abbreviations, etc.).

Similarly to Levenshtein Distance, Jaro-Winkler also tolerates scores below 100, which in turn are compared by n-Grams, with the help of Cosine. The idea is to use the Jaro-Winkler with thresholds flexible enough to filter out some of the false positives, without eliminating the true positives that we are less certain about. To remove the doubts about which are the real true positives, n-Grams is applied, aided by Cosine, in order to refine the dubious results obtained from Jaro-Winkler.

When using n-Grams, an initial comparison is made with the street types, which must be equal, otherwise the matches are discarded. This decision was made taking into account that there may be streets with the same name, but different street types (**Rua** de Santo Amaro vs**Travessa** de Santo Amaro, e.g.). After this check, the other strings are divided into tokens, then into bigrams in n-Grams, which in turn are transformed into vectors to allow these to be measured with Cosine. If more than half of these tokens have a score

below a defined threshold, the match is rejected. Generally, only strings with less than one or two tokens different from their matches are accepted, at most, which in our scope is seen as a spelling error. In order to test the behavior of each of the similarity measures, the thresholds considered for each of the measures is presented in Table 10.

Table 6.2: Thresholds considered for testing similarity measures.

	Levenshtein	**Jaro-Winkler**	***N*-Grams/Cosine**
Scenario #1	**70**	90	75
Scenario #2	**80**	90	75
Scenario #3	**90**	90	75
Scenario #4	80	**85**	75
Scenario #5	80	**95**	75
Scenario #6	80	90	**50**
Scenario #7	80	90	**90**

In the set of experiments, we tried to evaluate the performance of the similarity metrics that were explored throughout this thesis by varying the similarity thresholds for each measure individually, Being an ongoing project, tests of this kind had been done before, especially at the beginning, since previously only the Levenshtein Distance was used for measuring similarity, only the data with a score of 100 were considered correct, since they are the ones that are considered equal by the Levenshtein Distance. As such, median thresholds were applied to all similarity measures in order to maintain flexibility between the amount of false positives and true positives obtained.

6.3 Experimental Results

This section details the experimental methodology and the results obtained in each requirement. To avoid repeating too much information to be compared, we consider only buildings for comparison in these tests, since the street name is what is being considered for comparison, and also since the goal of the tests is to prove the executability of the approach defined throughout the thesis. Each subsection corresponds to the results obtained for each of the requirements, sorted in the same order as laid out in Chapter 3.

6.3.1 Results for R1

To test the defined approach with the similarity measures chosen for this requirement, 16067 records were considered. The partial zip code chosen was 8200, since it had the most geographic coordinates within all partial zip code subsets. The lack of geographic coordinates is a problem that Altice has tried to solve before, but without much success. Table 6.3 shows the results in terms of precision for this experiment.

Table 6.3: Precision for each similarity measure used in R1's first step.

	Levenshtein	**Jaro-Winkler**	***N*-Grams/Cosine**	**Overall**
Scenario #1	30.84	3.33	52.17	96.17
Scenario #2	35.89	3.47	80.00	98.92
Scenario #3	58.57	4.11	66.67	98.90
Scenario #4	35.89	2.70	80.00	98.92
Scenario #5	35.89	8.80	78.57	98.92
Scenario #6	35.89	3.47	75.00	98.57
Scenario #7	35.89	3.47	62.50	98.90

Addresses that get a 100% similarity score using Levenshtein Distance are always considered to be correct (true positives), and those below that threshold (false positives) are processed by the remaining algorithms. That is, the number of false positives that are obtained at Levenshtein Distance are also passed through Jaro-Winkler and n-Grams, affecting their precision percentage, due to the thresholds considered for these when Levenshtein's threshold is relatively low. The number of false positives decreases as it goes through the algorithms and as the thresholds considered vary. The percentages are so low because it is relative to the total matches found by Levenshtein Distance against the matches that were actually included with Jaro-Winkler and n-Grams filtering, which makes these results a bit misleading at first glance. The result that really matters in percentage terms is the one represented in the last column of Table 6.3, which corresponds to the overall precision when using the similarity measures together.

6.3.2 Results for R2

To test the defined approach with the similarity measures chosen for this requirement, 2540 records were considered. The partial zip code chosen was 3030, because it is a subset that corresponds to the city of Coimbra, and it was thought it would be interesting to analyze a known data set for more dubious data like the POLARIS tickets. Table 6.4 shows the results in terms of precision for this experiment.

Table 6.4: Precision for each similarity measure used in R2.

	Levenshtein	**Jaro-Winkler**	***N*-Grams/Cosine**	**Overall**
Scenario #1	83.30	8.33	14.44	94.33
Scenario #2	85.92	8.05	13.79	94.46
Scenario #3	89.17	6.08	10.71	94.45
Scenario #4	85.92	7.04	13.79	94.46
Scenario #5	85.92	8.78	12.79	94.46
Scenario #6	85.92	8.05	9.79	90.85
Scenario #7	85.92	8.05	7.41	94.44

Unlike the results shown for R1, in Table 6.4 we can already see some consistency in terms of percentages. This is mostly due to the whole universe within the 8200 zip code being considered, and not just those without geographic coordinates. Since most of the true positives were found mostly by the Levenshtein Distance, the other similarity measures just filter out the false positives. These are mostly cases that have the same street name, but different street types, which in turn get high scores because most of the address is the same. As mentioned in Section 6.2, these cases are considered as false positives by the measures. Given that we are dealing with tickets, and many of these tickets being aggregated into POLARIS are from various external sources other than the ones being used

in these experiments, the consistency of this data is somewhat low. Some of these problems are in the way the data is laid out, sometimes repeating information when normalization of this data is done by POLARIS processes. Nevertheless, regardless of the intermediate results obtained in the similarity measures, the overall precision is still above 90%, which is a good indicator that the approach really works.

6.3.3 Results for R3

To test the defined approach with the similarity measures chosen for this requirement, 2905 records were considered. The partial zip code chosen was 5090, because it had the most geographic coordinates within all partial zip code subsets. Table 6.5 shows the results in terms of precision for this experiment.

Table 6.5: Precision for each similarity measure used in R3.

	Levenshtein	**Jaro-Winkler**	***N*-Grams/Cosine**	**Overall**
Scenario #1	85.68	19.78	38.93	96.44
Scenario #2	88.53	20.70	38.58	96.52
Scenario #3	92.06	20.75	35.54	96.51
Scenario #4	88.53	18.03	38.58	96.52
Scenario #5	88.53	16.23	25.71	96.48
Scenario #6	88.53	20.70	25.73	93.48
Scenario #7	88.50	20.70	31.50	96.50

Table 6.6: Precision for each similarity measure used in R1's last step.

	Levenshtein	**Jaro-Winkler**	***N*-Grams/Cosine**	**Overall**
Scenario #1	78.82	21.95	57.14	96.36
Scenario #2	83.77	25.71	72.73	98.15
Scenario #3	90.45	33.33	72.73	96.95
Scenario #4	83.77	25.00	72.73	98.15
Scenario #5	83.77	23.81	57.14	98.10
Scenario #6	83.77	25.71	37.50	91.43
Scenario #7	83.59	25.00	66.67	96.32

To complete the results presented, the precision obtained by the similarity measures to search for new addresses, which corresponds to the final step of the approach defined for R3, is presented in Table 6.6. In this step only the data from CTT and INE are considered, since the goal is to consolidate the addresses that are present in both sources for POLARIS.

6.3.4 Results for R4

To test the defined approach with the similarity measures chosen for this requirement, 2905 records were considered. It was thought that it would be interesting to see the behavior of the measures with varying the sources considered for the same cases, although the approach is equal to R3 to some extent, so the partial zip code chosen was 5090, as was R3. Table 6.7 shows the results in terms of precision for this experiment.

Table 6.7: Precision for each similarity measure used in R4.

	Levenshtein	**Jaro-Winkler**	***N*-Grams/Cosine**	**Overall**
Scenario #1	78.22	38.95	72.63	95.45
Scenario #2	83.96	42.29	83.13	96.97
Scenario #3	92.65	56.41	82.89	97.07
Scenario #4	83.96	39.11	78.72	96.23
Scenario #5	83.96	68.67	79.41	96.90
Scenario #6	83.96	42.29	82.22	96.73
Scenario #7	83.96	42.29	81.58	96.94

Similar to R3, here only the information between POLARIS and CTT is crossed, and the obtained results for this experiment prove the good quality of the data present in CTT, as the precision was higher for all similarity measures than in any of the other experiments done. This is due to the fact that the CTT source has an alternative table that contains synonym information for each of its streets, which the other sources do not have. As such, the goal is to inherit the synonyms from the corresponding CTT streets that contain this information.

6.4 Conclusion

The obtained results show that, when properly tuned, the different similarity metrics can achieve very similar results in terms of matching quality, when used together. Although we have relied on Python implementations for the different similarity metrics, which indeed could be further optimized, the values that are reported already provide a good indication of the executability of these approaches for consolidation purposes by Altice.

While no experiments were formulated with thresholds significantly lower than those presented in the tests, these would bring disastrous results at the level of number of wrong matches. This is because a lower threshold implies that the similarity measures perform more operations to transform one address into another.

Based on the analysis done for the tests, Scenario #2 with thresholds of 80 for the Levenshtein Distance, 90 for the Jaro-Winkler Distance and 75 for the n-Grams/Cosine, are the most appropriate for the generality of cases. This is due to the fact that they present a higher overall than the other results obtained in this study. Also, because of the way the similarity measures are aligned, another reason why thresholds of Scenario #2 are chosen is because of the need to make the Levenshtein Distance more flexible, so as not to reject matches that are correct but are discarded because they have different arrangements or even contain errors. The Jaro-Winkler running with a higher threshold than Levenshtein, the input matches become more limited, giving more confidence, being finally validated by n-Grams and Cosine, to ensure that it is the same address, through the separation and comparison of the words contained in the addresses in tokens. For a better understanding of the percentages shown in each of the results presented above refer to Appendix C, which presents the same tables, but with concrete numbers, where the separation between true positives vsfalse positives is also made for each similarity measure.

Chapter 7

Conclusion and Future Work

This chapter presents the conclusions regarding the results achieved with the study focused on the consolidation of address information and accommodating the functionalities developed for this purpose in an operable software that allows Altice's autonomous use. To achieve Altice's proposed objectives, in-depth research on the topic and work related to information consolidation and extraction was required in order to build a customized approach to address the problem at hand, since there was no specific approach to solve it.

To get a better understanding of the project objectives and build the customized approach, the requirements and constraints were defined together with Altice, in order to understand the steps and decisions to be taken towards achieving the project's goals. With the requirements and constraints defined, there was a further need to study the data to understand how they are arranged, how they intersect, and how to group the data. As the customized approach was being designed, the researched similarity measures that best fit the problem were applied, which match an input string representing an address with the targeted string of the same type in the database. Moreover, steps such as normalizing the data have been taken in order to standardize the information across all sources and decrease the possibility of missing addresses with good information when using the similarity measures.

After all functionalities were implemented, based on the approach, a proposal was designed for the creation of the service with those functionalities. Following software engineering standards, an architecture for the system was defined in order to detail how the system will be designed, as well as to seek Altice's approval more easily. This architecture involves the creation of an API that contains the endpoints for each of the requests intended by the requirements, which will serve to communicate the responses to the requests executed by a user without having to deal with the server directly.

Finally, in order to validate the feasibility of the designed approaches, test scenarios were prepared to evaluate which thresholds to use for each of the similarity measures, in order to have some flexibility about possible correct addresses, but at the same time have enough confidence in the matches produced. All requirements were successfully addressed and it was possible to get a good response from each one of them. A good metric for this success, is that at this moment Altice has already managed to consolidate a considerable portion of the data found through the approaches produced and demonstrated in this document.

7.1 Future Work

The work carried out in this dissertation presented promising results, and showed that it is possible to continue this work to integrate natural language processing techniques to solve problems related to address consolidation. As future work, the system has a lot of maturing to do in terms of ways to deal with differences in the layout of the addresses between sources, which brings us back to the normalization problems. The data normalization problems have various levels of difficulty, such as the inclusion of accentuation, Roman numerals, numbers expressed in digits or in full, etc. Some of these problems mentioned are solved by removing all special characters (including accentuation) directly in all sources. Other problems are solved with string tokenization, which allows individual analysis of each word included in the address, and normalization is done using fixed lists of abbreviations and common Portuguese connectors. Another challenge is related to spelling error detection, that although basic spelling error detection is already in place, there is still room for improvement.

The direction of future work is in this sense, where machine learning models could be used, in order to understand where the errors in the trained models are occurring and to study how these challenges can be tackled.

References

[1] Extração, transformação e carregamento (etl) - azure architecture center. https://docs.microsoft.com/pt-pt/azure/architecture/data-guide/relational-data/etl. [Accessed 23-12-2021].

[2] FuzzyWuzzy: Fuzzy String Matching in Python - ChairNerd. [Accessed 20-12-2022].

[3] Python Record Linkage Toolkit Documentation — Python Record Linkage Toolkit 0.14 documentation. [Accessed 11-01-2022].

[4] What is data consolidation?

[5] Richmond Alake. Understanding Cosine Similarity And Its Application.

[6] Srividya K. Bansal and Sebastian Kagemann. Integrating big data: A semantic extract-transform-load framework. *Computer*, 48(3):42–50, 2015.

[7] Raman Bhadauria. ETL Process in Data Warehouse. https://www.geeksforgeeks.org/etl-process-in-data-warehouse/. [Accessed 27-12-2021].

[8] Simon Brown. Software architecture for developers-volume 2: Visualise, document and explore your software architecture. *Ebook*, 2017.

[9] Yunbo Cao, Zhiyuan Chen, Jiamin Zhu, Pei Yue, Chin-Yew Lin, and Yong Yu. *Leveraging Unlabeled Data to Scale Blocking for Record Linkage.*

[10] Fernando Casanovas Martín. *Approximate string matching algorithms in art media archives.* PhD thesis, UPC, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, May 2009.

[11] Sam Comber and Daniel Arribas-Bel. Machine learning innovations in address matching: A practical comparison of word2vec and crfs. *Transactions in GIS*, 23(2):334–348, 2019.

[12] Johannes Dannelov. Study on Record Linkage regarding Accuracy and Scalability. page 28, 2082.

[13] Alexandre Veríssimo de Sousa Marinho. Approximate string matching and duplicate detection in the deep learning era. pages 5–8, 2018.

[14] DeepAI. Cosine Similarity.

[15] Maria Del, María Angeles, and Adrian Espino-Gamez. Comparison of methods hamming distance, jaro, and monge-elkan. 05 2015.

[16] Halbert L. Dunn. Record Linkage. 36(12):1412–1416.

[17] IBM Cloud Education. What is ETL (Extract, Transform, Load)? [Accessed 24-06-2022].

[18] Ivan P. Fellegi and Alan B. Sunter. A Theory for Record Linkage. 64(328):1183–1210.

[19] Kavita Ganesan. What are N-Grams? [Accessed 05-08-2022].

[20] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: Theory, practice & open challenges. 5(12):2018–2019.

[21] David Hand and Peter Christen. A note on using the f-measure for evaluating record linkage algorithms. *Statistics and Computing*, 28, 05 2018.

[22] Md. Mosabbir Hossain, Md. Farhan Labib, Ahmed Sady Rifat, Amit Kumar Das, and Monira Mukta. Auto-correction of english to bengali transliteration system using levenshtein distance. In *2019 7th International Conference on Smart Computing and Communications (ICSCC)*, pages 1–5, 2019.

[23] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[24] Kjytay. What is Jaro/Jaro-Winkler similarity? [Accessed 29-07-2022].

[25] Karen Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24:377–439, 1992.

[26] Brinardi Leonardo and Seng Hansun. Text documents plagiarism detection using rabin-karp and jaro-winkler distance algorithms. *Indonesian Journal of Electrical Engineering and Computer Science*, 5:462–471, 02 2017.

[27] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals.

[28] Alexandre Marinho. Approximate string matching and duplicate detection in the deep learning era. 10 2018.

[29] Richard Milne. Jaro winkler. `https://github.com/richmilne/JaroWinkler`. [Accessed 20-12-2022].

[30] Faisal Rahutomo, Teruaki Kitasuka, and Masayoshi Aritsugi. Semantic cosine similarity. 10 2012.

[31] Rui Santos, Patricia Murrieta-Flores, and Bruno Martins. Learning to combine multiple string similarity metrics for effective toponym matching. *International Journal of Digital Earth*, 11(9):913–938, 2018.

[32] Snowflake. What is ETL (Extract, Transform, Load)? [Accessed 24-06-2022].

[33] Erkki Sutinen and Jorma Tarhio. On using q-gram locations in approximate string matching. pages 327–340, 09 1995.

[34] David Taylor. ETL (Extract, Transform, and Load) Process in Data Warehouse.

[35] Esko Ukkonen. Approximate string-matching and the q-gram distance. In Renato Capocelli, Alfredo De Santis, and Ugo Vaccaro, editors, *Sequences II*, pages 300–312, New York, NY, 1993. Springer New York.

[36] Jiannan Wang, Guoliang Li, and Jianhua Fe. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *2011 IEEE 27th International Conference on Data Engineering*, pages 458–469, 2011.

[37] Lidong Wang. Heterogeneous data and big data analytics. *Automatic Control and Information Sciences*, 3(1):8–15, 2017.

[38] Yaoshu Wang, Jianbin Qin, and Wei Wang. Efficient approximate entity matching using jaro-winkler distance. pages 231–239, 10 2017.

[39] Rahmadi Wijaya and Bambang Pudjoatmodjo. An overview and implementation of extraction-transformation-loading (etl) process in data warehouse (case study: Department of agriculture). In *2015 3rd International Conference on Information and Communication Technology (ICoICT)*, pages 70–74, 2015.

[40] William Winkler. The state of record linkage and current research problems. *Statist. Med.*, 14, 10 1999.

[41] Minghe Yu, Guoliang Li, Dong Deng, and Jianhua Feng. String similarity search and join: a survey. *Frontiers of Computer Science*, 10, 11 2015.

Appendices

This page is intentionally left blank.

Appendix A

The following appendices correspond to the documents provided by Altice for the treatment of abbreviations related to street types and titles, as well as the file containing the output format for each of the proposed requirements.

Abbreviations for street types

Abrev.	Indicativo	Abrev.	Indicativo
ACSS	Acesso	IMP	Impasse
ADRO	Adro	IC	Itinerário Complementar
AL	Alameda	IP	Itinerário Principal
ALDT	Aldeamento	JRD	Jardim
ARRM	Arruamento	LAD	Ladeira
ATLH	Atalho	LG	Largo
AE	Auto-estrada	LEV	Levada
AV	Avenida	LOTEAM	Loteamento
AZ	Azinhaga	MTE	Monte
BR	Bairro	PASS	Passeio
BC	Beco	PQ	Parque
CAIS	Cais	PTO	Pátio
CC	Calçada	PTE	Ponte
CCNH	Calçadinha	PC	Praça
CAM	Caminho	PCT	Praceta
CM	Caminho Municipal	PROLNG	Prolongamento
CV	Caminho Vicinal	QLHA	Quelha
CPO	Campo	QTA	Quinta
CAN	Canada	RAM	Ramal
CSL	Casal	RAMP	Rampa
CTR	Centro	ROT	Rotunda
CID	Cidade	ROSS	Rossio
CRC	Circular	R	Rua
CRCV	Circunvalação	RLA	Ruela
CRZT	Cruzamento	SIT	Sítio
EMPR	Empreendimento	TAP	Tapada
ENC	Encosta	TERR	Terreiro
ENT	Entrada	TRANSV	Transversal
ENTRC	Entroncamento	TV	Travessa
ESC	Escadas	URB	Urbanização
ESCNH	Escadinhas	VAR	Variante
ESTR	Estrada	VR	Vereda
EM	Estrada Municipal	VIA	Via
EN	Estrada Nacional	VDTO	Viaduto
ER	Estrada Regional	VLA	Viela
FTE	Fonte	VL	Vila
GAV	Gaveto	ZN	Zona
HRD	Herdade		

Abbreviations for street titles

Abrev.	Título
AB	Abade
ACT	Actor/Actriz
ALC	Alcaide
ALF	Alferes
ALM	Almirante
ARC	Arcebispo
ARQ	Arquitecto/Arquitecta
ASP	Aspirante
AVD	Aviador
BR	Barão
BTA D	Beata Dona
BTO	Beato
BPO	Bispo
BPO D	Bispo Dom
BB	Bombeiro
BRG	Brigadeiro
CB	Cabo
CAP	Capitão
C MOR	Capitão-Mor
C TEN	Capitão-Tenente
CAR	Cardeal
CAR D	Cardeal Dom
COM	Comandante
CMD	Comendador
CMR	Comodoro
CD D	Conde Dom
CD	Conde/Condessa
CON	Cónego
CONS	Conselheiro
CNSUL	Cônsul
C ALM	Contra-Almirante
COR	Coronel
CORR	Corregedor
DEPUT	Deputado/Deputada
DSB	Desembargador
DSP	Despachante
D	Dom/Dona
DR JZ	Doutor Juíz
DR	Doutor/Doutora
DQ	Duque/Duquesa
EMB	Embaixador
ENF	Enfermeiro
ENG	Engenheira/Engenheiro
ENG D	Engenheiro Dom
EGT	Engenheiro Técnico
ESCRT	Escritor/Escritora
ESC	Escultor/Escultora
FR	Frei
FR D	Frei Dom
FUR	Furriel
GEN	General

Abrev.	Título
GD	Guarda
GDJ	Guarda-Jóias
GDM	Guarda-Mor
INF	Infante
INF D	Infante Dom/Infanta Dona
INT	Intendente
JZ	Juíz
J CNS	Juíz Conselheiro
MTO	Maestro
MAJ	Major
MCH	Marechal
MAQ	Marquês/Marquesa
MESTR	Mestre
MINIS	Ministro
MONS	Monsenhor
OPR	Operário
PE	Padre
PATR	Patrão
PILOT	Piloto
PINT	Pintor/Pintora
POETA	Poeta/Poetisa
PR	Presidente
PCB	Primeiro Cabo
PTEN	Primeiro Tenente
PRI D	Príncipe Dom
PRI	Príncipe/Princesa
PRIOR	Prior
PF DR	Professor Doutor
PF EG	Professor Engenheiro
PROF	Professor/Professora
RA	Rainha
RA D	Rainha Dona
RA ST	Rainha Santa
REI	Rei
REI D	Rei Dom
REV	Reverendo
ST	Santo/Santa
S	São
S FR	São Frei
SG	Sargento
SEC	Secretário/Secretária
SEN	Senador
SD	Soldado
TEN	Tenente
TEN C	Tenente-Coronel
TEN G	Tenente-General
TEN M	Tenente-Médico
V ALM	Vice-Almirante
VIG	Vigário
VISC	Visconde/Viscondessa

Format defined for the output files

O POLARIS dentro dos parâmetros do projeto necessitará de outputs da UC abaixo descritos para reutilização dos processos atuais.
Quaisquer alterações aos formatos implica esforço de alteração do lado POLARIS e com impacto em sistemas consumidores da informação.

- **Ficheiro de novas moradas a certificar**
  Os campos deste ficheiro de input não deverão ter qualquer separador e deverão ser de tamanho fixo.

Campo	Formato	Comprimento	Observações
REFERENCIA	Alfanumérico	20	Referência para a morada. Deverá ser preenchido pelo sistema origem com um valor gerado para cada morada de acordo com a seguinte regra: ***T***<*nnn*><*pppppppppppppppp*> : em que, <nnn> será um código numérico de 3 dígitos atribuído pelo Mestre de Moradas a cada sistema consumidor de moradas (ver Tabela de códigos de sistemas utilizadores de moradas); <pppppppppppppppp> será uma string de 16 caracteres gerada por cada sistema e única dentro de cada sistema.
TIPO_PEDIDO_CERTIFICACAO	Alfanumérico	1	Tipo de Pedido de Certificação. Pode ser { "**A**", "**P**" , "**Q**", "**U**", "**C**", "**I**", "**X**"} onde, "**A**" – Arruamento; "**P**" – Prédio; "**Q**" – Prédio com Código Postal; "**U**" – Unidade Alojamento; "**C**" – Unidade de Alojamento com Código Postal e Giro

Format defined for the output files

Campo	Formato	Comprimento	Observações
			Postal; "**I**" – Internacional; "**X**" – Apartado.
INDICATIVO_ARRUAMENTO	Alfanumérico	5	Logradouro
TITULO	Alfanumérico	5	Título do Arruamento
DESIGNACAO_ARRUAMENTO	Alfanumérico	100	Designação do Arruamento
PRI_IND	Alfanumérico	12	Primeiro indicador de morada (vulgo "Nº de Polícia") Limitado a 11 posições no caso de o indicador ser numérico ou Abreviatura + numérico
SEG_IND	Alfanumérico	8	Segundo indicador de morada (vulgo "Andar") Limitado a 7 caracteres no caso de o indicador ser numérico ou Abreviatura + Numérico
TER_IND	Alfanumérico	8	Terceiro indicador de morada (vulgo "Fracção") Limitado a 7 caracteres no caso de o indicador ser numérico ou Abreviatura + Numérico
CODIGO_LOCALIDADE	Alfanumérico	11	Código de Localidade Formato: DDCCFFLLLLL
NOME_LOCALIDADE	Alfanumérico	40	Nome da Localidade Ignorado caso o Código de Localidade venha preenchido
REGIAO	Alfanumérico	20	Nome da Região da morada internacional
PAIS	Alfanumérico	20	Nome do País da morada internacional
ESTAB_POSTAL	Alfanumérico	40	Estabelecimento Postal do Apartado
NRO_APARTADO	Numérico	5	Número do Apartado
CP4	Numérico	4	Código Postal
CP3	Numérico	3	Giro Postal
OBSERVACOES	Alfanumérico	50	Informação adicional de morada
ID_MOR_ARR_ENVIADO	Alfanumérico	20	**Obsoleto** Id de morada correspondente ao Arruamento Nota: Este campo, quando enviado, deve ser numérico
ID_MOR_PRD_ENVIADO	Alfanumérico	20	**Obsoleto** Id de morada correspondente ao Prédio Nota: Este campo, quando enviado, deve ser numérico

Format defined for the output files

Campo	Formato	Comprimento	Observações
ID_MOR_UAL_ENVIADO	Alfanumérico	20	**Obsoleto** Id de morada correspondente à UAL Nota: Este campo, quando enviado, deve ser numérico
DT_HR_MORADA_ENVIADA	Alfanumérico	26	**Obsoleto** Data/Hora da morada quando foram enviados os campos de ID anteriores. Este campo deve ter o seguinte formato: AAAA-MM-DD-hh.mm.ss.nnnnnn
NR_PISOS	Alfanumérico	2	Nº de pisos do prédio Nota: Este campo, quando enviado, deve ser numérico
FRACC_RESIDENCIAIS	Alfanumérico	3	Nº de fracções residenciais existentes no prédio Nota: Este campo, quando enviado, deve ser numérico
FRACC_COMERCIAIS	Alfanumérico	3	Nº de fracções comerciais existentes no prédio Nota: Este campo, quando enviado, deve ser numérico
FRACC_ESCRITORIOS	Alfanumérico	3	Nº de fracções para escritórios existentes no prédio Nota: Este campo, quando enviado, deve ser numérico
OUTRAS_FRACC	Alfanumérico	3	Nº de fracções que não sejam residenciais nem comerciais nem escritórios. Nota: Este campo, quando enviado, deve ser numérico
MARGEM_ERRO	Alfanumérico	6	Margem de erro em centímetros Nota: Este campo, quando enviado, deve ser numérico
X_LONG	Alfanumérico	20	Ordenada X (em metros) ou Longitude em graus decimais. Nota: Este campo, quando enviado, deve ser numérico no seguinte formato: {'-','+',' '}ddddddd.ddddddddd d – dígito Separador decimal: Ponto ('.') Sinal: Mais, Menos, Espaço ('+','-',' ')
Y_LAT	Alfanumérico	20	Ordenada Y (em metros) ou Latitude em graus decimais. Nota: Este campo, quando enviado, deve ser numérico no seguinte formato: {'-','+',' '}ddddddd.ddddddddd d – dígito Separador decimal: Ponto ('.') Sinal: Mais, Menos, Espaço ('+','-',' ')
ID_EXTERNO_MORADA	Alfanumérico	25	Identificador externo da morada.

Campo	Formato	Comprimento	Observações
FONTE_GEOREF	Alfanumérico	5	Fonte de Georreferenciação. Valores possíveis: 2243 – BDI 2245 – CTT 2236 – Digitalização SIG 2235 – GPS 2999 – INE 2244 – Levantamento GPON 558 – Outro 2805 – SIGNET 2731 – SITED Nota: Este campo, quando enviado, deve ser numérico
SISTEMA_COORDENADAS	Alfanumérico	5	Sistema de Coordenadas. Valores possíveis: 2237 – DATUM 73 2370 – UTM Fuso 25 - Gr. Ocidental Açores 2371 – UTM Fuso 26 - Gr. Oriental e Central Açores 2372 – UTM Fuso 28 – Madeira 2238 - WGS84 Nota: Este campo, quando enviado, deve ser numérico.
ID_PROJETO	Alfanumérico	32	Apenas enviado por Netwin. Restantes sistemas enviam sem este campo.
ID_SURVEY	Alfanumérico	16	Apenas enviado por Netwin. Restantes sistemas enviam sem este campo.
CODIGO_OPERADOR	Alfanumérico	3	Identificador do operador associado ao id externo.

- **Ficheiro de Unidades de Alojamento a fundir.** Dois campos de 20 carateres separados por ";":

Campo	Formato	Notas
Id_morada_origem	char(20)	Identificador Polaris da morada a fundir
id_morada_destino	char(20)	Identificador Polaris da morada que prevalece após fusão

Format defined for the output files

Será com base neste ficheiro que fundiremos a unidade alojamento origem na unidade de alojamento destino.

- **Criação de Arruamentos**: ficheiro com o seguinte formato (campos separados por ";"):

Campo	Formato	Notas
COD_LOCALIDADE	varchar(11)	Código de localidade do arruamento
NOME_LOCALIDADE	varchar(40)	Nome da localidade do arruamento
ID_ARRUAM_PROVISORIO	char(20)	Id provisório para fazer mapeamento entre o input e o output
IND_ARRUAM	varchar(5)	Abreviatura do indicativo do arruamento segundo a tabela de valores de referência já fornecida.
TIT_ARRUAM	varchar(5)	Abreviatura do título do arruamento segundo a tabela de valores de referência já fornecida.
DESIG_ARRUAM	varchar(100)	Designação do arruamento
TIPO_PARIDADE_CP7	varchar(40)	Valores possíveis: Troço de portas Impares Troço de portas Pares Troço de portas Sequenciais Porta com outra identificação Código Postal de Grande Cliente (nenhum)
INDIC_INFERIOR	varchar(12)	Indicador inferior do intervalo de portas ao qual se aplica o CP7
INDIC_SUPERIOR	varchar(12)	Indicador superior do intervalo de portas ao qual se aplica o CP7
CP4	char(4)	Código Postal
CPL3	char(3)	Giro Postal
DESIG_POSTAL	varchar(40)	Designação Postal

Exemplo:
31020200000;Curral das Freiras;PTST0000000000000001;IM;;1 Estrada da Capela;Porta com outra identificação;;;9030;322;CURRAL DAS FREIRAS
31020200000;Curral das Freiras;PTST0000000000000002;EC;;Pau Formoso;Porta com outra identificação;;;9030;352;CURRAL DAS FREIRAS
31020200000;Curral das Freiras;PTST0000000000000003;IM;;1 Estrada do Colmeal;Porta com outra identificação;;;9030;325;CURRAL DAS FREIRAS

- **Criação de Sinónimos de Prédio**: ficheiro com o seguinte formato (campos separados por ";"):

Campo	Formato	Notas

Operação	Varchar(50)	Valor fixo: CRIAR_SINONIMO_PREDIO
Id_morada	Varchar(20)	Identificador Polaris do prédio oficial
Pri_ind	Varchar(12)	Indicador de morada do sinónimo a criar

Exemplo:
CRIAR_SINONIMO_PREDIO; P0000000000000123853;LT 20

- **Criação de Sinónimos de Arruamento**: ficheiro com o seguinte formato (campos separados por ";"):

Campo	Formato	Notas
Operação	Varchar(50)	Valor fixo: CRIAR_SINONIMO_ARRUAMENTO
Id_morada	Varchar(20)	Identificador Polaris do arruamento oficial
Ind_arr	Varchar(5)	Abreviatura do indicativo de arruamento (R, AV, AL, TV, ...). Ver tabela de referência.
Tit_arr	Varchar(5)	Abreviatura do título do arruamento (DR, PROF, CAP, ...). Ver tabela de referência.
Nome_Sinónimo	Varchar(36)	Nome do arruamento sinónimo

Exemplo:
CRIAR_SINONIMO_ARRUAMENTO;A0000000000000012345;R;DR;ANTÓNIO SEABRA

This page is intentionally left blank.

Appendix B

Análise de Agrupamento e Intersecção

Para descrever a abordagem analítica aplicada nas intersecções, começamos por apresentar as intersecções consideradas e as variáveis que serviram de base à definição dos critérios de intersecção. De seguida, descrevemos as restrições que foram usadas em cada critério ou variável e os resultados obtidos com as intersecções.

Descrição dos casos considerados

Tendo em conta as 4 tabelas principais tabelas, nomeadamente **Alojamentos (CTT)**, **Inspire (INE)**, **Survey** e **Polaris**, foram realizados os 6 cruzamentos possíveis entre estas (duas a duas). Nestes cruzamentos as variáveis consideradas foram: o **código do postal** e as coordenadas de **latitude** e de **longitude**, de maneira a que fosse exequível obter o maior número possível de correspondências, tanto para descobrir possíveis melhorias como permitir a consolidação e adição de informação nova que não esteja presente Polaris.

Quando possível, outras variáveis como o **número de porta, localidade, código de localidade**, etc., são utilizados para dar mais confiança aos *clusters* resultantes das intersecções.

De modo a manter a coerência dos valores que se seguem, são expostas na Tabela 1 todas as bases de dados consideradas, bem como os números associados a cada uma delas, divididas por tipo.

Table 1: Sumário das bases de dados disponibilizadas.

Fonte de dados	**Total**	**# Prédios**	**# Alojamentos**	**Observações**
Polaris (certificada)	13 283 056	4 908 027	7 746 882	Apenas moradas certificadas
Polaris (*tickets*)	17 316 773	1 216 848	16 032 352	Moradas em processo de validação (não certificadas)
Survey	3 028 995	3 028 995	-	Só contém moradas de prédios
CTT	6 887 615	4 496 338	4 496 338	Prédios obtidos a partir das unidades de alojamento
INE	5 910 282	3 566 268	5 910 282	Prédios obtidos a partir das unidades de alojamento

Variáveis de intersecção

Em termos de comparações, no caso das coordenadas, foi feita com base no arredondamento à 4ª casa decimal (correspondente a 11,132 metros de precisão) e na **distância euclidiana**, dado que a comparação direta entre elas era impossível pois a precisão varia entre as tabelas.

Para tal, estas duas abordagens foram utilizadas em termos analíticos para conseguir obter um bom número de correspondências e visualizar nos restantes valores que podem remeter para a consolidação da Polaris.

Também foi tida em consideração uma análise entre todas as tabelas utilizando os dois campos referidos anteriormente juntamente com o número da porta de cada uma de modo a extrair *clusters* com um maior nível de confiança, diminuindo ligeiramente o volume de correspondências. Para contrariar esta limitação, foi baixada a precisão da distância euclidiana de 11,132 para 22,264 metros, de maneira a abranger uma gama maior de moradas, mas mantendo a consistência de correspondências com o número da porta.

Para diminuir a redundância em termos de correspondências (por exemplo um prédio pode ter vários alojamentos o que origina correspondências duplicadas), foi ainda realizada uma separação das tabelas só com moradas de prédios para evitar *clusters* com muitas repetições, dada que a Polaris apresenta claramente esta distinção entre prédios e alojamentos. Para auxiliar a informação presente na Tabela 1, é mostrado na Tabela 2 como foi feita esta separação na Polaris.

Table 2: Segmentação da Polaris por tipo de dados.

Tipo de registos	Total de registos
Moradas c/ sinónimos	37 669
Arruamentos s/ sinónimos	420 810
Prédios s/ sinónimos	4 759 690
Unidades de alojamento	7 563 771
Internacionais e apartados	162 885

De referir ainda que 1 451 270 dos prédios sem sinónimos não possuem coordenadas, ou seja, só 69,51% destes é que contam com esta informação. Nenhuma das unidades de alojamento tem coordenadas diretas, mas como todas apontam para um prédio principal (campo id_predio), estas podem ser obtidas a partir dos seus prédios. Tal como há prédios sem coordenadas, o mesmo acontece com as unidades de alojamento, que contam com 5 726 973 dos 7 563 771 registos com estes campos preenchidos, isto é, 75.72%.

Resultados para a Análise de Agrupamento e Intersecção

Nesta secção apresentamos os resultados preliminares para a parte de análise de agrupamentos. Apesar de os parâmetros da abordagem como as distâncias limite e que outros atributos considerar nos casos em que as coordenadas estejam ausentes ainda estarem a ser afinados de acordo com as avaliações em andamento, os resultados já permitem mostrar que a abordagem é promissora, como discutido nas secções seguintes.

A. Polaris - Survey

Apesar da dimensão da base de dados Survey em termos de registos e campos, com base nas análises feitas, foram obtidas boas correspondências, validando a sua utilização. Nesta intersecção foram considerados os códigos postais, latitude e longitude. Tendo em conta que a Survey corresponde a moradas de prédios, foram considerados apenas os prédios presentes na Polaris.

De um modo geral, baseando a análise dos datasets com a distância euclidiana e o arredondamento à 4ª casa decimal, foi possível incluir mais correspondências entre as tabelas utilizando a distância euclidiana do que propriamente o arredondamento, sendo que em termos percentuais obteve-se cerca de 71,61% da Survey e 48,52% da Polaris com a distância euclidiana, contra 53,82% e 35.05%, respetivamente, dos arredondamentos, fazendo com que esta última análise seja, de certa forma, inviável.

Com base nisto, observam-se bastantes correspondências corretas, sendo que em alguns casos há erros devido à proximidade entre moradas. Para contornar este problema, manipulou-se o campo local_e presente na Survey, que contém o nome da rua e o número de porta juntos, apenas separados por um carater, de modo a retirar apenas o número de porta e utilizá-lo como campo de comparação. Com isto, as percentagens também diminuem,

o que é justificável pelo facto de incluir este campo para reduzir a margem de erros de proximidade, dando mais confiança que as correspondências obtidas são as mesmas. Ao fazer esta comparação apenas com os prédios resultaram 53.11% e 34.90% da Survey e da Polaris, respetivamente.

É possível retirar e adicionar novos prédios à Polaris, visto que existem códigos-postais do lado da Survey que não existem na Polaris, por exemplo. Porém, em termos de complexidade de tabelas, a Survey possui pouca informação acerca destes prédios, devido ao número reduzido de campos em relação à Polaris. Para tal, há a possibilidade de complementar estes dados com informação de outras tabelas, como a Inspire (INE) e a Alojamentos (CTT). Em termos de melhorias, logo que haja uma correspondência fidedigna, há a possibilidade de melhorar alguma informação, como por exemplo a adição de GPS em prédios da Polaris que não contenham este campo preenchido.

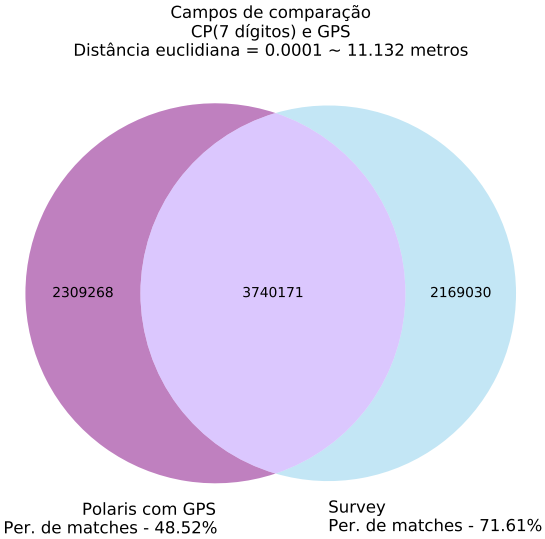

Figure 1: Intersecção entre moradas **Polaris - Survey** baseado no CP7 e distância euclidiana.

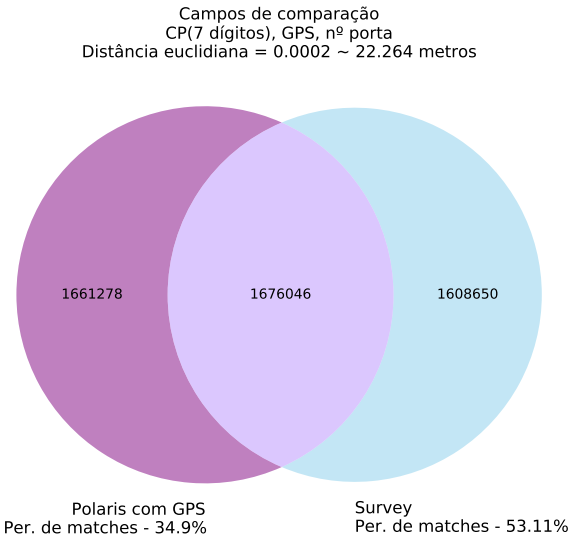

Figure 2: Intersecção entre moradas **Polaris - Survey** baseado no CP7, número de porta e distância euclidiana.

B. Polaris - Inspire (INE)

Face aos testes realizados anteriormente, aqui já é feita a completa distinção entre prédios e unidades de alojamentos de ambas as tabelas. Esta distinção é feita com o intuito de melhorar a resultante em termos de correspondências por parte dos prédios, dado que como os estes têm o mesmo GPS para cada uma das suas unidades de alojamento e sendo este um dos campos principais utilizados a título de comparação, ao cruzar todas as unidades de alojamentos, gera uma série de repetições entre estas, caso não haja um campo que ajude a distinguir entre cada uma.

Assim, sabendo que a morada e o GPS são iguais para todas as unidades correspondentes, evita-se este problema considerando apenas os prédios, o que simplifica o processo de identificação entre moradas equivalentes e moradas a adicionar.

Identificados os prédios, é possível extrair informação sobre as unidades que se encontram nestes, tais como o piso, número da porta e lado, visto serem campos presentes na Inspire. Em alguns casos, poder-se-á incluir o número de pisos, logo que haja a possibilidade de fazer esta distinção com confiança para não induzir em erros. Dado que a localização do INE é fidedigna, poder-se-á substituir os valores com zero (sem coordenadas) no lado da Polaris e indicar a fonte de georreferenciação.

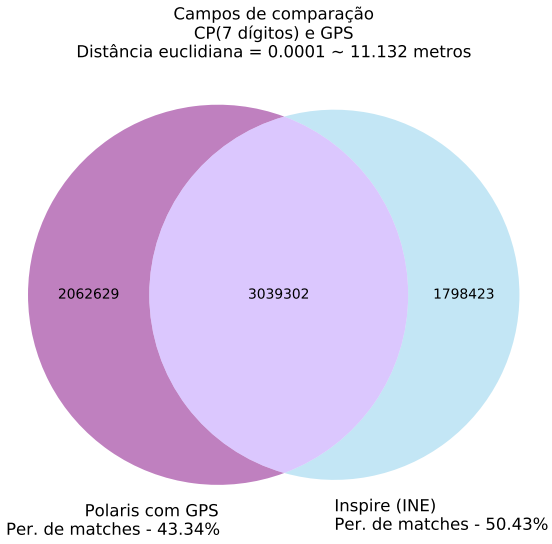

Figure 3: Intersecção entre moradas **Polaris - INE** baseado no CP7 e distância euclidiana.

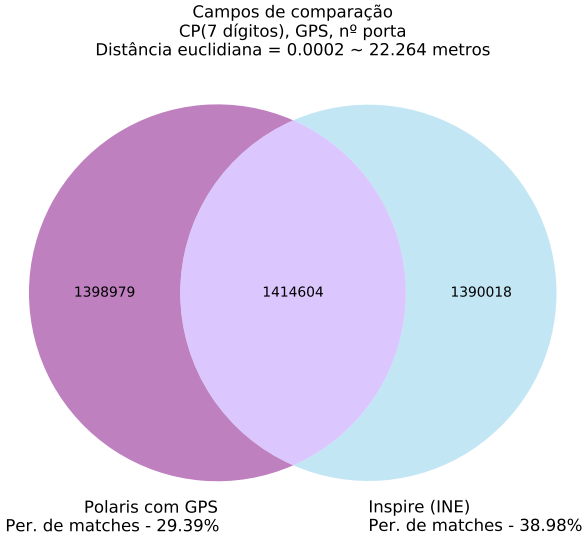

Figure 4: Intersecção entre moradas **Polaris - INE** baseado no CP7, número de porta e distância euclidiana.

C. Polaris - Alojamentos (CTT)

Dando continuidade à análise anterior e mantendo a mesma abordagem para todas as bases de dados, aqui também foi aplicada a caracterização entre prédios e unidades de alojamento para contornar as duplicações.

Tendo a base de dados dos CTT o mesmo grau de relevância que a do INE, esta análise é feita quase que em paralelo, devido a terem alguns campos em comum, logo, com base na identificação predial, é possível extrair informação sobre as unidades que se encontram nestes, tais como o piso, número da porta e lado, visto serem campos também presentes nos CTT. Em alguns casos, poder-se-á incluir o número de pisos, logo que haja a possibilidade

de fazer esta distinção com confiança para não induzir em erros. Considerando que as coordenadas dos CTT possuem a limitação de estarem identificadas ao nível da caixa postal, verificando e considerando o valor da distância euclidiana no cruzamento dos CTT com as outras tabelas, será fazível a determinação e correção, se for o caso, das coordenadas em falta no lado da Polaris.

Face aos números obtidos, apesar de serem notavelmente distantes comparativamente ao cruzamento entre Polaris e Inspire (INE) em termos percentuais, as moradas que se encontram do lado de fora deste podem ser aproveitadas, logo que estas não existam no lado da Polaris, sendo esta verificação auxiliada pela similaridade textual.

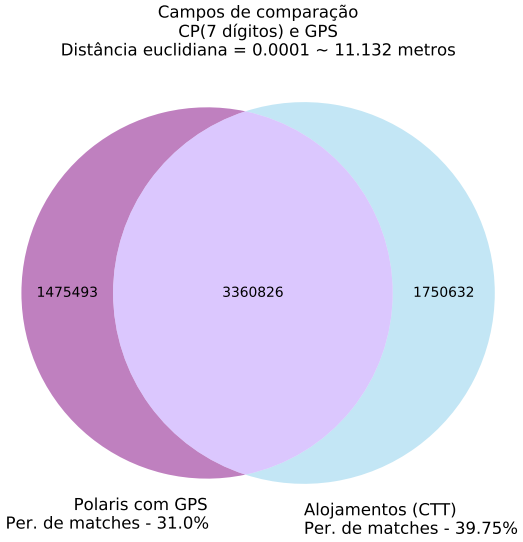

Figure 5: Intersecção entre moradas **Polaris - CTT** baseado no CP7 e distância euclidiana.

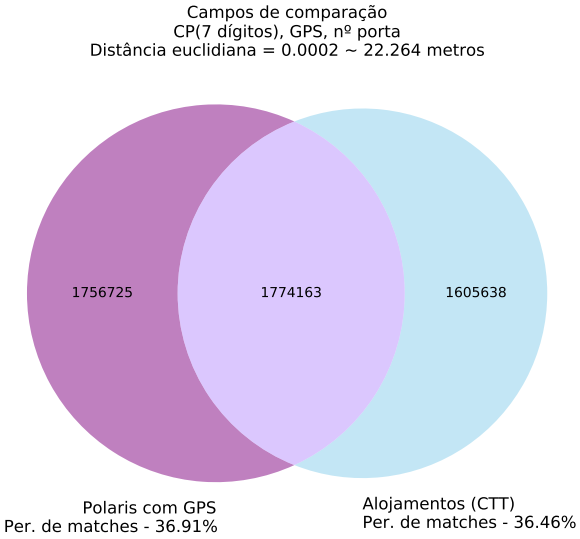

Figure 6: Intersecção entre moradas **Polaris - CTT** baseado no CP7, número de porta e distância euclidiana.

D. Inspire (INE) - Alojamentos (CTT)

Sendo as duas tabelas mais semelhantes e relevantes em termos de campos de comparação e de unidades de alojamento, respetivamente, dado que a Survey é limitada no que toca às unidades dos prédios.

Este cruzamento é feito com o intuito de averiguar a fidedignidade das duas bases de dados e onde se diferenciam, o que é importante, visto que tendo as coordenadas GPS dos CTT a limitação de estarem identificadas ao nível da caixa postal, quando estas discrepâncias forem corretamente reconhecidas, poder-se-á utilizar este cruzamento para melhorar e contornar as imprecisões associadas.

No que diz respeito às correspondências, as percentagens mostraram-se igualmente semelhantes, o que viabiliza a utilização de ambas as tabelas como fontes de dados a utilizar para a consolidação para a Polaris.

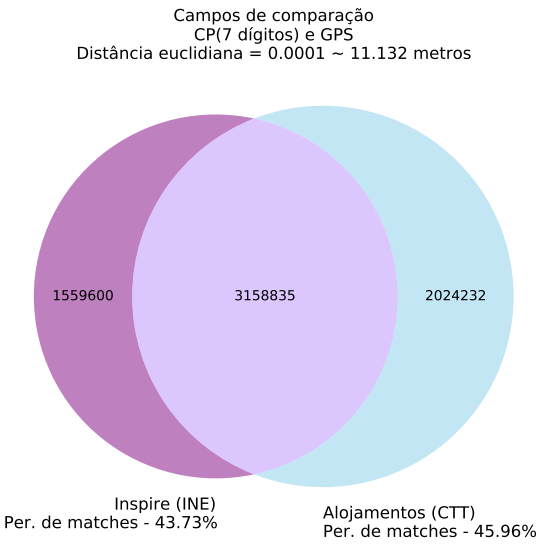

Figure 7: Intersecção entre moradas **INE - CTT** baseado no CP7 e distância euclidiana.

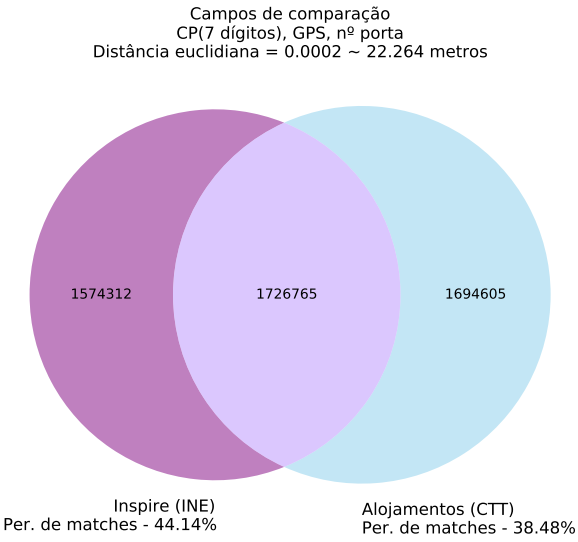

Figure 8: Intersecção entre moradas **INE - CTT** baseado no CP7, número de porta e distância euclidiana.

E. Inspire (INE) - Survey

Uma vez que a Survey não possui informação sobre as unidades de alojamento dos prédios que tem, esta poderá ser utilizada como reforço à base de dados dos INE a nível de coordenadas GPS e correção de nomes de arruamentos. No entanto, observando as percentagens obtidas, verifica-se que estas são semelhantes às do cruzamento entre Polaris e Inspire, o que reforça a ideia de que ambas as tabelas são de facto usáveis e pertinentes no que diz respeito tanto à consolidação, adição e remoção de moradas à Polaris.

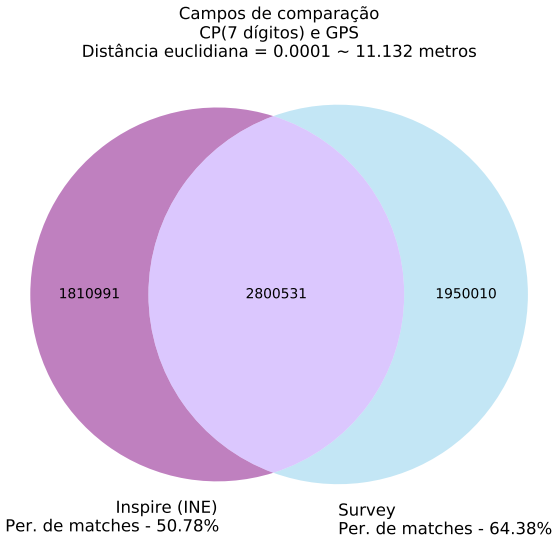

Figure 9: Intersecção entre moradas **INE - Survey** baseado no CP7 e distância euclidiana.

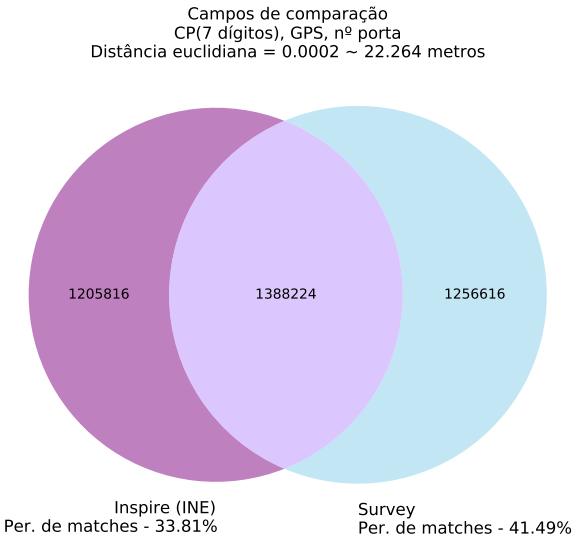

Figure 10: Intersecção entre moradas **INE - Survey** baseado no CP7, número de porta e distância euclidiana.

F. Alojamentos (CTT) - Survey

Apesar de serem duas tabelas completamente diferentes, era de esperar que houvesse um número de correspondências semelhante ao obtido no cruzamento com a tabela Inspire (INE), mas tendo em conta que o centróide do GPS da Survey se encontra ao nível da porta/edifício e o GPS dos CTT a nível da caixa postal, justifica-se assim o baixo número entre ambas.

Novamente, apesar de a Survey não possuir informação sobre as unidades de alojamento dos prédios que tem, esta poderá ser utilizada como complemento à base de dados do CTT a nível de coordenadas GPS e correção de nomes de arruamentos.

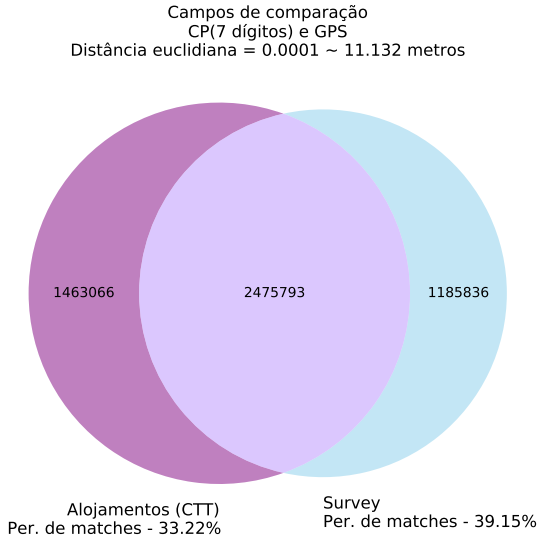

Figure 11: Intersecção entre moradas **CTT - Survey** baseado no CP7 e distância euclidiana.

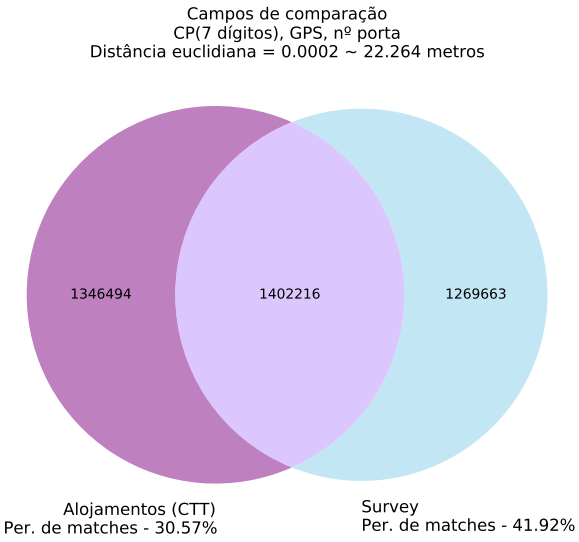

Figure 12: Intersecção entre moradas **CTT - Survey** baseado no CP7, número de porta e distância euclidiana.

Sobreposição vs *clusters* para várias distâncias

O objetivo de analisar este tópico é avaliar em que medida a oscilação da distância influencia a inclusão de correspondências, e, ao mesmo tempo o número de repetições para cada registo. Para tal, esta distância foi variada entre os 2.7 e os 55.6 metros, aproximadamente, de modo a termos dois extremos em que seja possível verificar qual o valor que permitam ter a maior eficácia, em termos inclusões e as repetições. Ainda assim, vai ser preciso nas próximas fases entender quão eficaz conseguem ser as técnicas de NLP, quando a operar sobre estes *clusters*.

Tendo em consideração a intersecção entre Polaris e Survey, e começando por fazer a apreciação da menor distância para a maior, é possível observar-se que, embora 2.7 metros se mostre promissor no que toca a ter um número baixo de repetições, ao mesmo tempo inclui poucos registos entre as duas tabelas. O mesmo acontece quando é o valor é elevado para 5.6 metros (metade do valor inicial considerado), que neste caso nota-se um salto significante no que diz a respeito à percentagem de inclusão, passando de **45%** para **62%** no caso da Survey, mas ao mesmo tempo continua a apresentar valores baixos. Nos 11.1 metros, olhando para o gráfico de barras da frequência absoluta, observa-se que com esta distância consegue-se obter um bom número de correspondências para um registo da Polaris, mas com isso é acarretado uma série de repetições, e isto deve-se à posição dos centróides das coordenadas GPS estarem localizados próximos uns dos outros e à precisão da distância ser relativamente mais baixa. Este fenómeno ocorre com mais frequência a partir dos 22.2 metros onde, ainda que continue a incluir mais alguns registos, nota-se um maior acréscimo de repetições em comparação ao seu antecessor, mas se tivermos em conta os gráficos das restantes distâncias, pode-se dizer que esta ainda é útil e muitas das repetições podem ser tratadas recorrendo à similaridade textual. Entre os 33.4 e os 55.6 metros, as conclusões serão as mesmas, dado que cada vez haverão mais repetições e mais correspondências, embora em termos de proporções, a primeira irá decolar com mais facilidade.

Baseados nestes números podemos observar que as distâncias mais promissoras são entre os 11.1 e os 22.2 metros, visto serem os valores em que se verifica um balanço aceitável entre o número de correspondências obtidas e o número de repetições, sem que o último aumente descontroladamente.

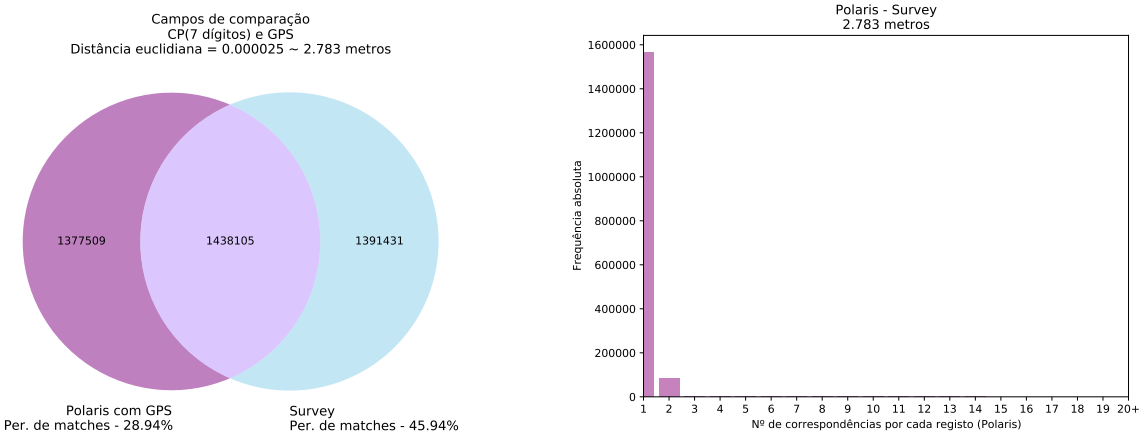

Figure 13: Intersecção entre moradas **Polaris - Survey**, com distância máxima de *2.8m* e mesmo CP7, e a frequência absoluta dos tamanhos dos *clusters* correspondentes.

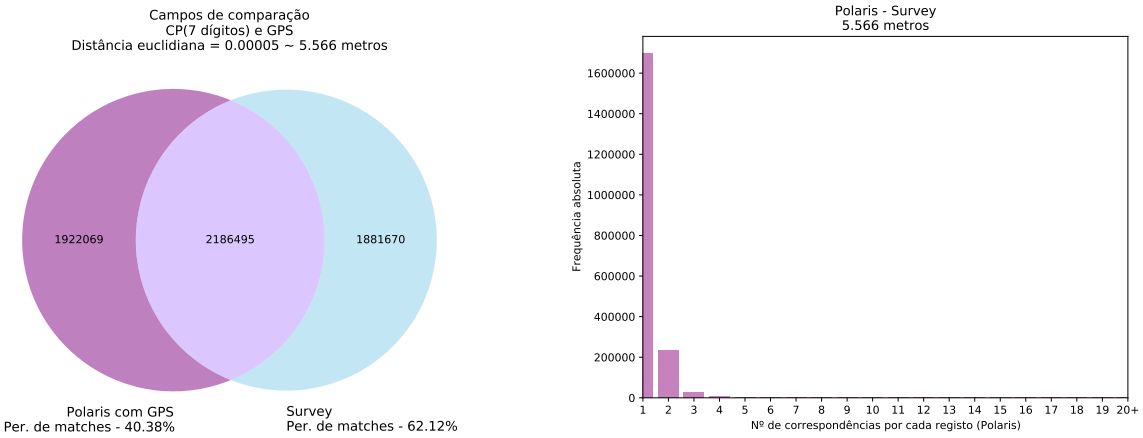

Figure 14: Intersecção entre moradas **Polaris - Survey**, com distância máxima de *5.6m* e mesmo CP7, e a frequência absoluta dos tamanhos dos *clusters* correspondentes.

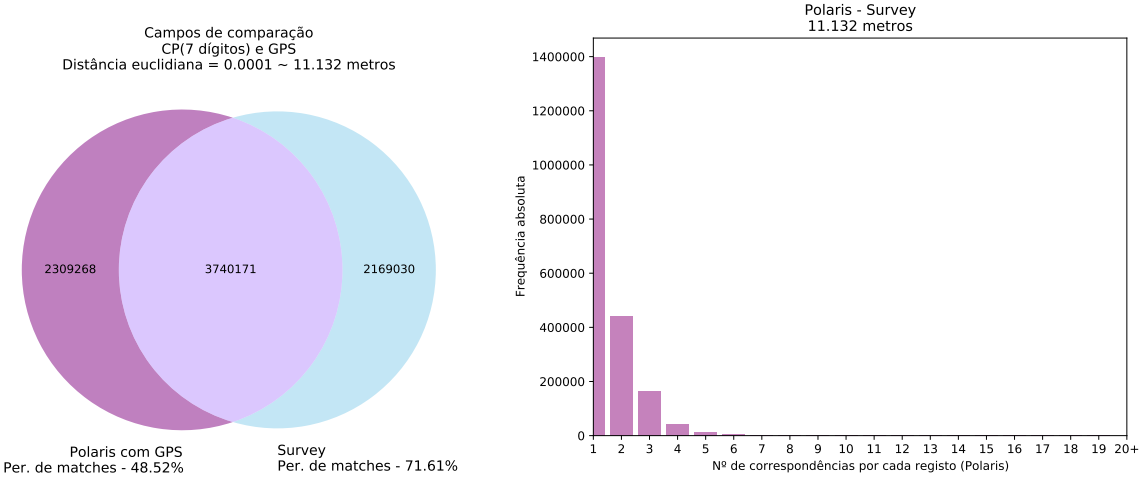

Figure 15: Intersecção entre moradas **Polaris - Survey**, com distância máxima de *11.1m* e mesmo CP7, e a frequência absoluta dos tamanhos dos *clusters* correspondentes.

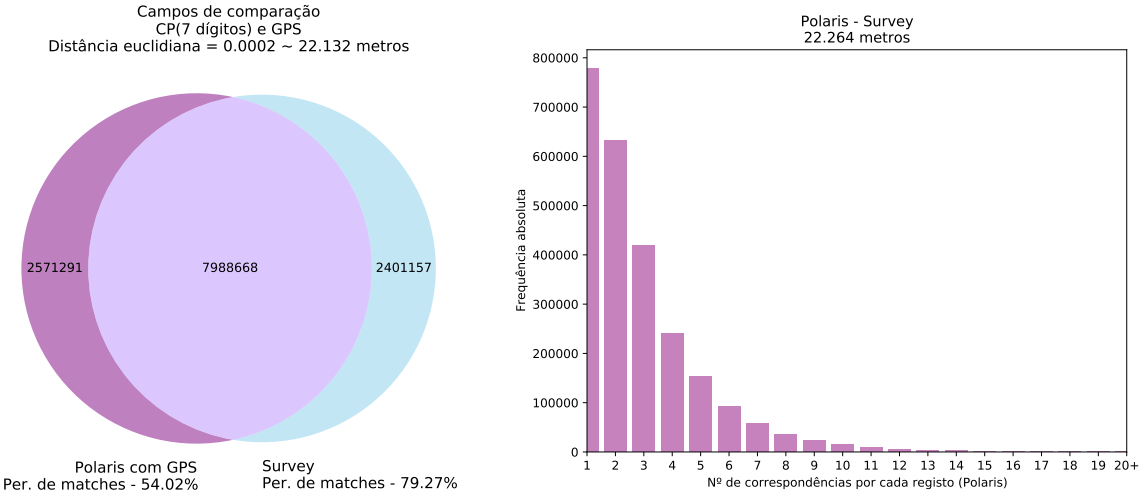

Figure 16: Intersecção entre moradas **Polaris - Survey**, com distância maxima de *22.2m* e mesmo CP7, e a frequência absoluta dos tamanhos dos *clusters* correspondentes.

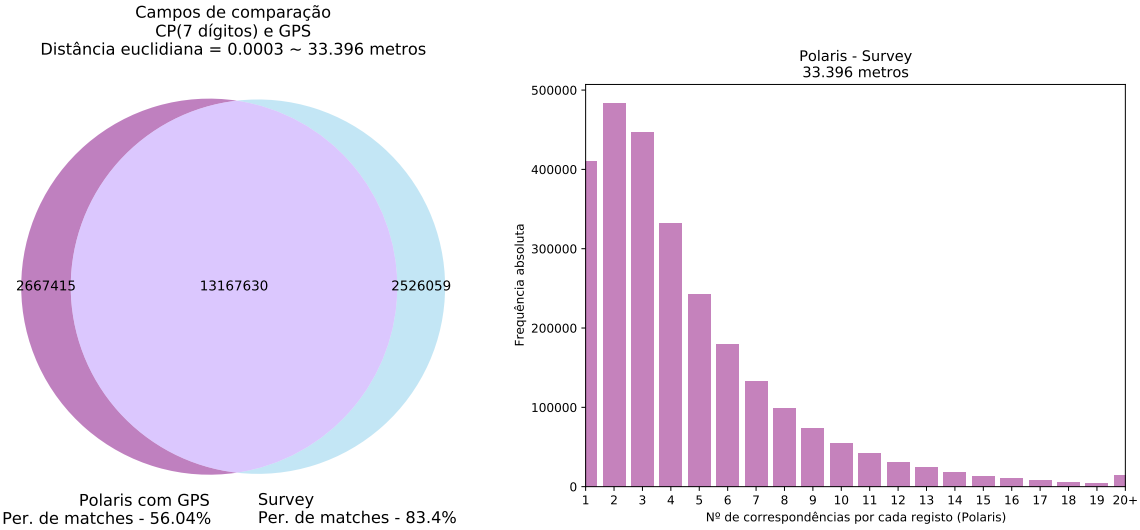

Figure 17: Intersecção entre moradas **Polaris - Survey**, com distância maxima de *33.4m* e mesmo CP7, e a frequência absoluta dos tamanhos dos *clusters* correspondentes.

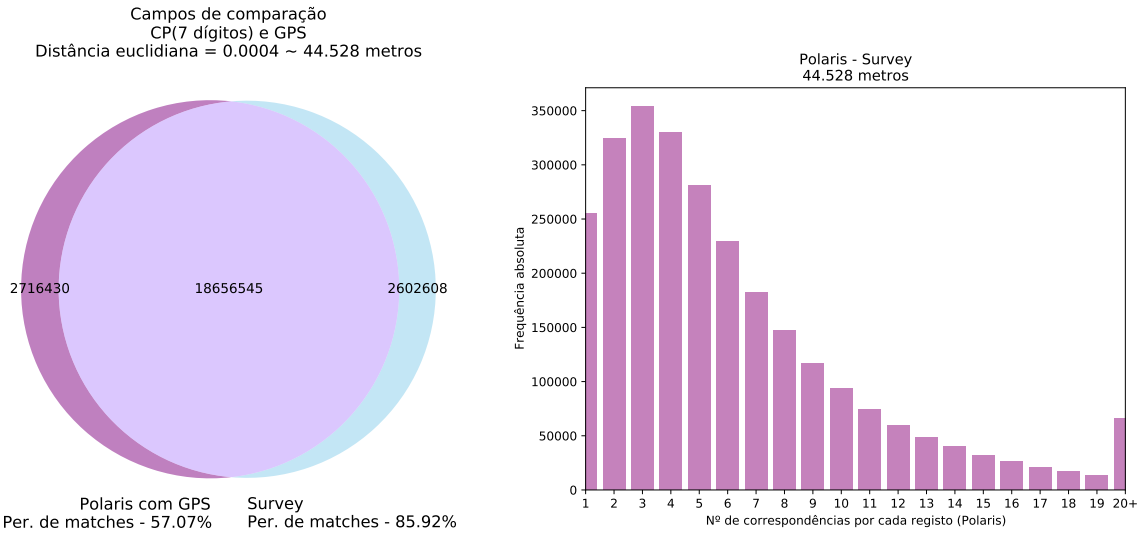

Figure 18: Intersecção entre moradas **Polaris - Survey**, com distância maxima de *44.5m* e mesmo CP7, e a frequência absoluta dos tamanhos dos *clusters* correspondentes.

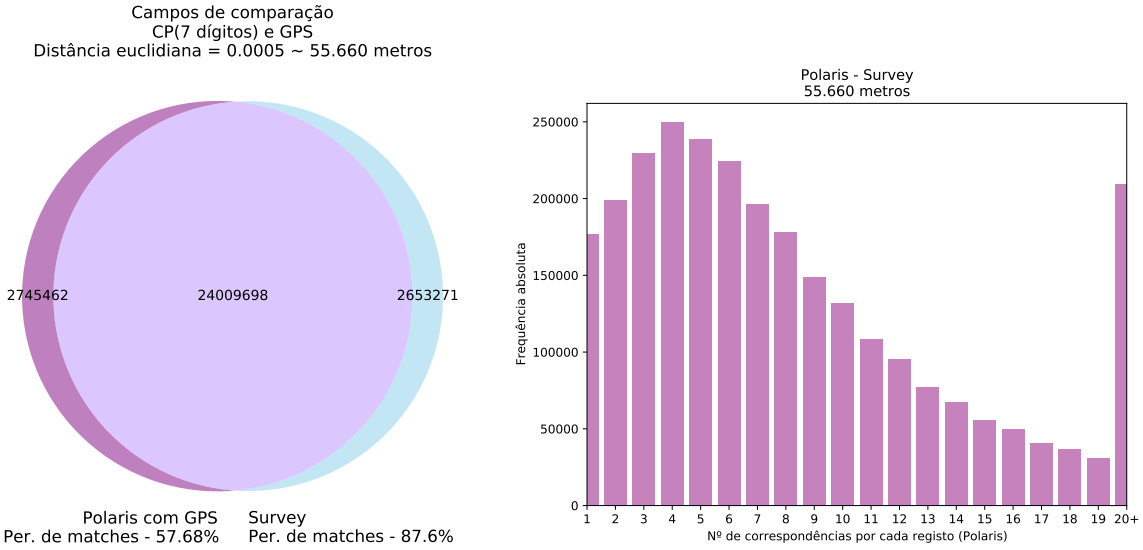

Figure 19: Intersecção entre moradas **Polaris - Survey**, com distância maxima de *55.6m* e mesmo CP7, e a frequência absoluta dos tamanhos dos *clusters* correspondentes.

Resultados

Dataset piloto

No sentido de validar as experiências feitas, o uso das ferramentas e técnicas descritas, foi proposto um *dataset* piloto com cerca de 50 mil registos vindos de Polaris, que conta com moradas certificadas e não certificadas (*tickets*), ambas relativas a Unidades de Alojamento, que representam habitações individuais. Assim sendo, o tratamento destes dados foi feito em direção ao primeiro requisito, que visa determinar coordenadas em falta na base de dados da Altice.

Para contextualizar, é mostrado na Tabela 3 o total de registos que são considerados para tratamento, separados por tipo de morada.

Table 3: Dispersão dos registos do *dataset* piloto

Tipo de morada	**Total de registos**
Certificada	15 449
Não certificada	34 620
Total	**50 069**

Para estes dados, através dos algoritmos de similaridade, foram encontradas um total de 19 119 equivalências nos 50 069 registos (∼38%).

Table 4: Total de equivalências encontradas no *dataset* piloto

Tipo de morada	Tipo correspondência	Total de registos	Percentagem
Certificadas	Única	9 039	47.2%
	Múltipla	251	1.3%
Não certificadas	Única	7 199	37.7%
	Múltipla	2 630	13.8%
Total	-	**19 119**	**100%**

Análise

De um modo geral, ao observar pela Tabela 4, podemos afirmar que os resultados obtidos foram bastante positivos, apesar de não resolver todos os registos incluídos no *dataset*. No entanto, tendo em conta que estamos a tratar dois tipos de moradas diferentes, faz sentido que estes sejam analisados separadamente e depois comparados para retirar conclusões. Esta separação é feita porque, apesar de terem formatos semelhantes, a qualidade da informação contida nas moradas certificadas é superior à das não certificadas, e isto reflete-se nos resultados obtidos.

Antes de partir para a análise aprofundada, deve ser feita também a separação feita entre **"Únicas"** e **"Múltiplas"**. A primeira acontece quando é encontrada apenas uma entrada para um registo em Polaris. A segunda acontece quando é encontrada mais do que uma entrada em todas as fontes para o mesmo registo em Polaris, e como tal, não é possível dizer com toda a certeza qual é a resposta correta. Para estes casos, há a necessidade de implementar um algoritmo de desempate entre as mesmas.

Em relação às moradas certificadas, nesta obtiveram-se 9 290 correspondências, sendo 9 039 com correspondências únicas e 251 com correspondências múltiplas, isto num total de 15 449 registos certificados, o que corresponde a cerca de 66% destes dados. Estes números mostram que os procedimentos trazem bons resultados, e que é exequível a aplicação das abordagens e técnicas definidas no contexto deste problema.

Em contraste, as moradas não certificadas (ou *tickets*), foram alcançadas 9 829 correspondências, sendo 7 199 com correspondências únicas e 2 630 com correspondências múltiplas num total 34 620 registos não certificados, o que corresponde a cerca de 24% destes dados. Estes últimos resultados geraram algumas dúvidas, tanto pelo baixo número de correspondências, como pela quantidade de correspondências múltiplas em relação às moradas certificadas, considerando que as percentagens entre os dois tipos de moradas foram visivelmente distantes.

Para melhor perceber o porquê da discrepância entre os dois tipos de morada e retirar conclusões, foi feita uma análise *à posteriori* apenas aos registos das moradas não certificadas. Nesta análise observou-se que alguns campos da morada destes dados não se encontram preenchidos ou totalmente normalizados (e.g. ter abreviaturas ou o número da porta agregado à morada), como normalmente é feito nas moradas certificadas. Estes detalhes dificultam a adaptação dos dados para comparação com os algoritmos de similaridade, resultando que os graus de similaridade sejam mais baixos, o que impede de garantir que se referem exatamente à mesma coisa. Ao mesmo passo que trará menos correspondências, poderá incluir também correspondências erradas em moradas que sejam muito semelhantes. Detetou-se ainda que cerca de 22 313 desses *tickets* não têm número de porta associado, tornando-os obsoletos, tendo em vista que este campo é imprescindível para ser atribuída a morada correta. Somados a estes, foram encontrados 4 650 casos com

código-postal parcialmente ou completamente vazio, o que exclui estes dados do agrupamento pelo código postal a 7 dígitos.

A partir dos primeiros resultados obtidos do *dataset* piloto, observa-se que a abordagem definida já se mostra auspiciosa. Embora não tenha sido possível dar resposta a todos os registos, foram alcançadas boas correspondências com grau de certeza muito elevado, tendo em conta as limitações associadas às moradas não certificadas. Como tal, para ter uma visão mais ampla do comportamento desta abordagem, partiu-se para um *dataset* de maior dimensão que não contém coordenadas geográficas, visto representar um dos grandes problemas da Altice da perspetiva da operacionalização desses mesmos dados.

Dataset com moradas certificadas

Tendo em vista a grande diferença entre certificadas e *tickets* nos primeiros resultados atingidos, os registos a ter em consideração para este teste são relativos apenas à Polaris certificada. Novamente, levando em consideração que os formatos são semelhantes, as moradas não certificadas carecem de alguma normalização na sua disposição, que por sua vez influenciará drasticamente os resultados obtidos. Neste sentido, consideram-se agora 1 365 706 registos sem GPS a serem comparados com as fontes auxiliares, através da similaridade textual. De referir ainda que estes são dados que a Altice já tinha tentado resolver anteriormente, e não foi possível dar resposta a todos. Os dados na Tabela 5 referem-se aos registos em que foram encontradas correspondências, tanto únicas como duplicadas, provenientes das outras fontes e que têm um GPS associado para consolidar para a Polaris.

Table 5: Total de correspondências encontradas para as moradas certificadas

Tipo de correspondência	**Total de registos**
Única	149 926
Múltipla	9 780
Total	**159 706**

Para estes dados, através dos algoritmos de similaridade, foram encontradas um total de 159 706 correspondências nos 1 365 706 registos sem coordenadas associadas (∼12%).

Análise

Nestas 159 706 equivalências, estão incluídas 9 780 múltiplas, que, apesar de não estarem a ser tratadas nesta fase, este número continua a mostrar-se bastante baixo quanto ao tratamento das certificadas, graças à prioridade definida inicialmente.

Do total, resta-nos ainda 149 926 registos com correspondências únicas, que são o foco destes resultados e os futuros dados a serem integrados na base de dados da Altice.

Se avaliarmos pelos resultados obtidos apenas nas moradas certificadas do *dataset* piloto explorado na Secção 7.1, em termos percentuais, os números foram visivelmente mais baixos. Isto acontece porque nem todos os dados em Polaris se encontram totalmente atualizados, tendo ainda alguns erros de atribuição associados ao código postal (e.g. 3000-350 e 3000-351, que correspondem à mesma rua, mas são referentes a moradas com números de porta pares e ímpares, respetivamente). Este problema influencia na fase de agrupamento dos dados, que nos mapeia erroneamente as moradas com as restantes fontes auxiliares.

Por sua vez, para os dados em que isto acontece, não serão encontradas correspondências, visto que o algoritmo de similaridade compara individualmente cada registo associado a um código postal. Acontece ainda que a base de dados, apesar de operacional, já tem alguns anos, e como existem ruas que evoluíram, mudaram de nome ou deixaram de existir com o passar do tempo, alguns registos podem corresponder a dados que já não existem nas fontes auxiliares.

Num próximo passo, tem-se como objetivo aplicar outras medidas no sentido de aumentar a margem destas correspondências. Para tal, uma das abordagens passa por fazer uma normalização a todos os campos da morada, e não somente o indicativo de arruamento e título associado (quando existe).

Em suma, apesar da percentagem obtida ser 12% contra 66% de correspondências para moradas certificadas no *dataset* piloto, os números continuam promissores e certamente trazem melhorias associadas, considerando que foram melhorados com sucesso 149 926 registos candidatos a adição de coordenadas geográficas.

Dataset com moradas não certificadas

Mantendo o mesmo processo de comparação produzido para os resultados das Secções 7.1 e 7.1, aplicou-se esta abordagem às moradas não certificadas, mas desta vez em direção ao requisito R2, que corresponde à certificação ou evolução de *tickets*. O objetivo nesta fase é de encontrar correspondências, no sentido de associar e compor as informações em falta nos registos não certificados. Para tal, desta vez consideram-se os dados da Polaris certificada como fonte auxiliar a procurar, com o intuito de validar, também, se estes *tickets* existem. Neste sentido, foram propostas 810 829 entradas não certificadas associadas a serviços da Altice, relativas a Unidades de Alojamento, que representam habitações individuais. De tal modo, estes dados foram agrupados novamente pelos seus códigos postais associados, e comparados através dos algoritmos de similaridade. Na prioridade anterior definida, o repositório de moradas certificadas passa a ter prioridade máximo, dada a afinidade entre os dados.

Para estes dados, através dos algoritmos de similaridade, foram encontradas um total de 184 693 correspondências nos 810 829 registos sem certificação (∼20%).

Table 6: Total de correspondências encontradas para as moradas não certificadas

Tipo de correspondência	**Total de registos**
Única	165 504
Múltipla	19 189
Total	**184 693**

Análise

Nestas 184 693 equivalências, estão incluídas 19 189 múltiplas, que, apesar de não estarem a ser tratadas nesta fase, este número, comparado aos resultados obtidos para as não certificadas na Secção 7.1, é relativamente mais baixo, e isto deve-se ao facto de serem *tickets* de moradas com serviços associados. Ao terem serviços associados, alguns destes dados já têm bastantes campos normalizados, embora muitos deles ainda estejam por tratar.

Do total, resta-nos ainda 165 504 registos com correspondências únicas, que são novamente

o foco destes resultados e os futuros dados a serem integrados na base de dados da Altice. De um modo geral, estes resultados continuam a não ser perfeitos, mas o processo continua a mostrar-se promissor na resolução de problemas deste tipo.

Aqui continuamos a ter o problema de alguns destes dados em consideração não se encontram totalmente normalizados e preenchidos. Acontece muitas vezes os campos da morada terem erros de entrada, ou até mesmo o código postal apenas preenchido pelos 4 primeiros dígitos. Tudo isto são problemas que influenciam negativamente tanto o agrupamento dos dados, como a comparação dos mesmos. Num próximo passo, tal como nos testes anteriores, tem-se como objetivo aplicar outras medidas no sentido de aumentar a margem destas correspondências. No entanto, a abordagem de normalização dos campos das moradas deste tipo é um processo mais delicado, visto que a informação, por vezes, não segue as regras comuns à base de dados certificada. Este problema deve-se ao facto de alguns destes dados serem derivados de moradas de outros sistemas.

Em suma, estes números ainda são bastante preliminares, mas já trazem algumas melhorias associadas aos 165 504 registos com correspondências, considerando que se conseguiria certificar completamente algumas destas moradas, ou pelo menos fazer com que estas mais informação, de modo a auxiliar no processo de certificação dentro dos sistemas da Altice.

This page is intentionally left blank.

Appendix C

Table 7: True Positives vsFalse Positives for R1

	Levenshtein		**Jaro-Winkler**		***N*-Grams/Cosine**		**Overall**	
	TP	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**
Scenario #1	276	619	12	348	12	11	276	11
Scenario #2	276	493	12	334	12	3	276	3
Scenario #3	270	191	6	140	6	3	270	3
Scenario #4	276	493	12	433	12	3	276	3
Scenario #5	276	493	11	114	11	3	275	3
Scenario #6	276	493	12	334	12	4	276	4
Scenario #7	276	493	12	334	5	3	269	3

Table 8: True Positives vsFalse Positives for R2

	Levenshtein		**Jaro-Winkler**		***N*-Grams/Cosine**		**Overall**	
	TP	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**
Scenario #1	754	210	74	116	69	26	735	35
Scenario #2	754	144	74	101	69	14	735	23
Scenario #3	744	59	66	51	63	13	729	22
Scenario #4	754	144	79	123	74	20	740	29
Scenario #5	754	144	57	26	54	14	720	23
Scenario #6	754	144	74	101	74	16	740	25
Scenario #7	754	144	74	101	62	14	728	23

Table 9: True Positives vsFalse Positives for R3

	Levenshtein		**Jaro-Winkler**		***N*-Grams/Cosine**		**Overall**	
	TP	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**
Scenario #1	2225	372	55	223	51	80	2219	82
Scenario #2	2223	288	53	203	49	78	2217	80
Scenario #3	2214	191	44	168	43	78	2211	80
Scenario #4	2223	288	53	241	49	78	2217	80
Scenario #5	2223	288	31	160	27	78	2195	80
Scenario #6	2223	288	53	203	53	153	2221	155
Scenario #7	2223	288	53	203	36	78	2204	80

Table 10: True Positives vsFalse Positives for R4

	Levenshtein		**Jaro-Winkler**		***N*-Grams/Cosine**		**Overall**	
	TP	**FP**	**TP**	**FP**	**TP**	**FP**	**TP**	**FP**
Scenario #1	1282	257	15	165	13	77	1280	77
Scenario #2	1281	210	14	160	12	75	1279	75
Scenario #3	1276	155	9	139	9	75	1276	75
Scenario #4	1281	210	14	185	12	75	1279	75
Scenario #5	1281	210	13	135	11	75	1278	75
Scenario #6	1281	210	14	160	14	12	1281	129
Scenario #7	1281	210	14	160	6	75	1273	75