FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Supervised Record Linkage For Banking Reconciliations

Simão Lopes Lúcio

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Professor João Pedro Mendes Moreira

Second Supervisor: Nuno Miguel Taveira Ribeiro

January, 2021

Supervised Record Linkage For Banking Reconciliations

Simão Lopes Lúcio

Mestrado Integrado em Engenharia Informática e Computação

January, 2021

Resumo

Num mundo em que a moeda e o seu valor são mais abstratos do que nunca, existe a necessidade the garantir que os systemas que organizam os contratos que realizamos todos os dias são fiáveis. A digitalização de, não só de dados genéricos mas, bens através dos quais regemos a nossa vida criou um novo problema: a garantia de consistência.

Se a contabilidade pessoal de um indivíduo está nas mãos de um sistema semi-autónomo, exposto a possíveis falhas, o mesmo é verdade para bancos de investimento como a Natixis. Linhas de negócio abertas em todo o mundo tornam a tarefa de manter a consistência algo muito difícil. As muitas variações em termos de arquitetura de sistemas, transversais a todas as plataformas de *trading* dos mercados mundiais, introduzem, potencialmente, pequenas anomalias nos dados que podem vir a traduzir-se em quantias significativas de dinheiro. Por este motivo, a Natixis implementou uma política de reconciliação de todas as transações realizadas pelo departamento de Corporate Investment Banking.

Este documento é um estudo do estado da arte e possivel implementação de uma *pipeline* de Record Linkage que, complementada por métodos supervisionados de *machine learning*, pode reduzir a probabilidade da introdução de erros humanos ao reconciliar as operações registadas por traders com sistemas externos. Esta é a primeira abordagem de uma implementação de um sistema mais robusto para comparação de registos de transações. Os resultados revelam-se promissores na mudança para um sistema de Record Linkage supervisionado, usando tecnicas de machine learning para classificação. Para uma única reconciliação, apenas uma pequena fração do conhecimento inicial acerca da estrutura de cada *data source* foi necessária para um resultado final de *true matching* acima dos 90%.

Abstract

In a world where currency and its value are more abstract than ever, there is a necessity for constant reassurance that the systems that keep the order in the contracts we register every day are trustworthy. The digitizing of, not only generic data but, goods with which we regulate our lives created a new problem: the guarantee of consistency.

If the personal bookkeeping of the everyday individual is in the hands of semi-autonomous systems not impervious to failure, the same is true for investment banks such as Natixis. Doing business across the world in multiple locations simultaneously makes the task of maintaining the consistency of the institutions records very hard. The many variations in system design across the trading platforms present in the open market, can introduce small data anomalies that represent very substantial sums of money. For that reason, Natixis implemented a policy of reconciling every transaction that comes from its Corporate Investment Banking branch.

This document is a study on the state of the art and a possible implementation of the record linkage pipeline that, coupled with supervised machine learning processes, can reduce the probability of human error when reconciling transactions made by traders with external systems records. It is the first approach to a possible implementation of a more autonomous and robust system of transaction records comparison. Results show a promising output when shifting from the current paradigm to record linkage pipelines with machine learning classification. For a single reconciliation, a significant reduction of the structure knowledge was needed to achieve a true matching rate of above 90%.

Acknowledgments

There are some key moments in our lives that kick-start the trajectory leading us to somewhere we always wanted to be, even if through places unknown. For me, the bet Gonçalo Sá, Daniel Costa, Nuno Ribeiro and José Eduardo Basto took on me, was one of those moments. Their support and friendship enabled me to keep putting my best foot forward in my professional life, against what is the *normal* path in which I never found my place. Without them, and Natixis, I would never be able to keep the dream I found in BEST Porto; to make the path my own and mold it into something of value to the people around me.

Nothing outside the norm is done without sacrifice, not only to me, but to the people that, somehow, feel inclined to make that same journey at my side. For the unconditional support, at all moments, and the calm kindness bestowed on me, I have to thank my girlfriend, Beatriz Pereira. If her place at my side was by choice, similar recognition has to be given to João Ferreira, which by sacred northern Portuguese bonds, and chance, was a big part of this chapter in my life and kept grounding me when ideas start to fly while, at the same time, keeping me on my toes every day.

While I navigated my own thoughts regarding an utopia of magical solutions, professor João Moreira was the compass that eventually led me back to firm soil. I thank him not only for that, but by the countless hours he endured, listening patiently, during the last year. This work would never be completed without his wisdom and the chance he took on me.

To close the circle of the story about his journey, one must look to the beginning of it all. My family, the source, are there, present, in every step. No distance or time can ever erase the profound marks every single one of them left in my being. All that I did, and will ever do, is owed to them. This is no exception. To my parents I leave a special place here, because of their infinite wisdom and the fact they never told me what to become, other than a stand up guy.

I reserve my final acknowledgment to my sister, who knows I'm *full of it*, but keeps loving me either way.

"Expose yourself to your deepest fear;
after that, fear has no power, and the fear of freedom shrinks and vanishes.
You are free."

Jim Morrison

Contents

List of Figures

List of Tables

Glossary

Reconciliation	Set of transactions with the same type of asset, market geolocation and other optional generic parameters
Trade	Transaction registered by a trader in a market
Source/Provider	External or internal system, either from the bank or a market, where trades are registered. It's also the origin of transaction files for reconciliation
RL	Record Linkage
Match	A pair of two representations of the same trade
True Match	A match without any issues regarding difference in values
User	End user of the reconciliation system. The user is tasked with finding and reporting issues in matches
Business Analyst	Works in the back-office of the system. His job is to tune the system to find matches so that users can evaluate possible issues with true matches

Chapter 1

Introduction

1.1 Data Reconciliations in the Banking World

The banking system of the XXI century relies on several layers of human and automated systems. On day-to-day personal interactions with commercial banks, many of the interfaces individuals are presented with in the form of ATM operations, home banking solutions or yearly banking reports are the result of multiple streams of data being compared and evaluated for errors or inconsistencies. Having an accurate representation of its financial flows is a constant worry and need of financial institutions. The same principle is true for the Corporate Investment Banking (CIB) branch of Natixis.

Natixis operates in every open market in the world, with prevalence in the European compared to the Asian and American ones - 70% and 30% transaction volume, respectively. Between them, the investments spread through: equities, fixed income and commodities. This represents a considerable volume of daily transactions by the entity, almost every day of the year.

The typical traders' portfolio can only be safely calculated if all of his transactions are accounted for. Although Natixis keeps records of all this, the sheer daily volume and number of different systems and platforms it uses makes the task of reconciling all of this data very hard. The process of registering a transaction usually involves human registry in a local and external system (in relation to the bank) that is performed by a third party staff member, not necessarily the trader that wishes to do so. Being that this process happens across the world, communication barriers, phonetic differences and different date/currency/number standards can induce errors into the system. If not, they can make the process of consolidating the central knowledge base of the banks' financial situation harder. That gap between the time an action is performed and, then, consolidated can represent significant losses in a world where the markets are very volatile.

Small inconsistencies in systems overtime, human error and external systems' abstractions are on the basis of the problem of reconciling the financial flow data. To overcome this, the CIB department of the bank created a team of analysts and developers targeted with the task of analyzing and verifying every transaction made by the traders. Inconsistencies found are then

communicated to the relevant parties and all the verifications are saved for future inspection by internal bank auditors.

1.2 Motivation

As stated in the previous section, 1.1, banks in general rely on very different information systems to store and manage their financial movements. This heterogeneity made the need for a data reconciliation team in the CIB department of Natixis in order to keep a record and communicate data inconsistencies to the people involved in the process. The team is responsible for analyzing streams of data from internal and external providers, cluster them by type (internally known as reconciliation), match the internal reference records with correspondent external comparison records and evaluate the match for discrepancies.

This article is focused in the integration and matching of the data. There are a lot of intermediate steps to this process. Maybe the most significant one being the data normalization and consolidation. At the time of writing this article, that whole section is based on a rule system. When a new reconciliation is added to the system, the two providers involved start appending the transaction data to their stream. The team is only given a set of generic information regarding the transactions that need to be evaluated. Many times, the streams are not structured resulting in very dirty data sets. Business analysts then go through the list of unmatched transactions and try to infer rules for data normalization and clustering that filter that new reconciliation specifically. The whole process takes a long time and the resulting rule knowledge base is not robust enough to face variations in reconciliation representation by the providers.

1.3 Objectives

From the start, it was established that reconciling data is an integral part of the international trading process. Natixis' goal, since the creation of a reconciliation team, has been to keep the banks' knowledge base current and accurate in order to make sure every trader's portfolio is correct and can be managed without issues. To achieve consistency and accuracy, two main aspects must be part of the system that takes care of that side of business: speed and robustness. In the current implementation the speed bottleneck is on the manual definition and update of the systems' data matching and integrating rules. It is based on the inference and manual input of the team. More than that, is based on the transmission of knowledge between two types of people - business analysts (BA) and developers, conceptualized by the first and implemented by the second - which can on itself be the origin of filtering errors of the data sources. The robustness is put to test and fails every time there is a change on the data stream. Either during the introduction of new reconciliations or caused by changes in previous data representation by the sources, there are unmatched or falsely matched transactions. Lack of flexibility to these changes also makes the speed of the process worse as every wrong match has to be manually evaluated and new rules or changes to new ones need to be done.

The purpose of this study is to evaluate the viability of the implementation of a record linkage pipeline backed by supervised classification in order to reduce the impact of human interaction with the system and remove the need of user created rules for data integration. The proposed system should take advantage of the data already accumulated by the team as part of the reconciliation process in the last years. This data set will be used to train supervised processes in order to create decision models for the matching status of transactions between sources. The final goal is to change the process in order to redirect the bank's users efforts.

1.4 Document Structure

This document describes the current state of methodologies regarding record linkage and supervised machine learning approaches that create added value to the process, followed by the specific problem description of trading data reconciliations in Natixis and a proposed study on the best framework to implement, in order to address issues at hand.

The report is divided in 5 chapters, the following 4 are:

- Chapter 2 — is an introduction to the record linkage methodology and terminology. It is divided into three sub-chapters that provide an introduction to record linkage, the definition of the record linkage problem and the pipeline of a generic record linkage system. For each module presented in the pipeline, several different methodologies are described in order to achieve the goal of integrating two different data sets.
- Chapter 3 — here, the usage of supervised machine learning methods in record linkage is divided into generic model generation, with considerations regarding record linkage applications, examples and description of the most successful supervised classification methods used in record linkage and other achievements in the integration of supervised machine learning in record linkage frameworks.
- Chapter 4 — this chapter describes the current reconciliation system, implemented in Natixis, from end to end, while enumerating the specifics of the issue addressed by this study.
- Chapter 5 — in the methodology chapter a definition of the scope of the solution, in the context of the problem, is presented, as well as the choices made regarding the process of implementing the supervised record linkage pipeline.
- Chapter 6 — the experiments and results chapter contains the results of the implementation of the previous chapter, as well as the corresponding discussion.
- Chapter 7 — in this chapter are enumerated the writers conclusions regarding the study of the Natixis proposal and future expectations for the proposed implementation.

Chapter 2

Record Linkage

2.1 Background in Record Linkage

The process of identifying records corresponding to the same entity, across different data sources, is commonly called record linkage (other nomenclatures to describe similar processes include - entity matching, entity heterogeneity, entity identification, object isomerism, instance identification, merge/purge, entity reconciliation, list washing, data cleaning [17]. From this point on, it will be referred to simply as RL. The process might refer to any type of data proposed to be analyzed, and the fields where it can be applied are only limited to the maximum distribution of records among the sources taken under consideration. The entities in question can include personal information from individuals, physical sensors, companies, products or any other entity that can be represented by a set of attributes on a data structure.

The data analyzed is from relevant sources for the field of study. Some of those might have an ideal structure such as well-defined relational databases with known attributes and ontology relations between them. This includes cases like internal databases for systems where all the business rules are present and transparent. Other cases might evaluate data coming from semi-structured or unstructured sources. The definition for a lack of structure is broad [13] but it might include any number of deficiencies in the representation of a single record and its relationship with the domain it is included on - lack of unique key identifiers, heterogeneous relationship definitions, misrepresentation of attributes or even lack of attributes definition. A good example of semi-structured data are the results of web scraping where there is information concerning the domain spread across HTML tags and paragraphs of text.

Different sets of data and their quality define the best RL approach to be used. The ideal scenario would be to have unique identifier keys that represent the same entity. Using those identifiers to match records gives an error-free result with exact correspondence. Although possible, it is not common to be in the presence of such well-structured sources. Many factors weigh in the degrading of data, either in its genesis, such as typographical errors or independent system changes overtime [25]. When unique key identifiers are not present, or they are not consistent between the data sources, one is in the presence of a probabilistic or machine learning supervised RL problem.

The first modern interpretation of RL, using a non-deterministic approach, can be tracked to Newcombe and Kennedy's 1959 article [41] and later formalization by Felligi and Sunter in 1962 [22]. Since then, numerous areas have taken advantage of a specific implementation of RL that fits the problem best, but no overall solution has been proposed that can encapsulate the full extent of variables included in the pipeline [32]. In this document, some options of probabilistic and machine learning based RL are going to be described.

2.2 RL Problem Definition

In RL the question is always about combining information that refers to the same individual or entity, spread through various sources. Taking the health sector, an area with a prevalent use of RL methods, as an example - if every registered health related activity during our lifetime can be matched to each individual, there are many real world applications and benefits to be extracted. The workload associated with such an endeavor on a case by case bases or manually is not feasible and can limit the size and type of the population to be studied. In cases like this, two different objectives, trough different types of RL, can be pointed to. Matching the individuals exactly can create a personal health board. While matching similar individuals regarding certain constraints (partially), can help on the creation of a population level picture of certain groups of people as well as use the same data in new studies [15][24].

In the present article, the problem being defined is aligned with the second case. Matching two transaction records using a deterministic or exact approach will leave out cases in which there are significant errors in the data. During the reconciliation process the objective is to find possible errors related to manual introduction or degradation of values in the systems being reconciled. While looking for an exact match, the method would leave out possible matches in which one source presents one of two things: different representation of the same data (if the data normalization process leaves some attributes out) and incorrect values. That would defeat the purpose of trying to create an automatic system for matching the data in the first place. To solve such problems, instead of exact or deterministic RL, one is looking to a solution that implements statistical RL. In statistical RL one does not look to find the exact same information replicated in the sources, but the most likely match. This preserves the ability to manually evaluate the transaction values for errors, even if their representation is not exactly the same and the RL system cannot give a 100% verified match [15].

Let us say there are two sources - **A** and **B**. Each with - n_a and n_b records, respectively. This means that there are $n_a \times n_b$ possible matches. Other than that, there is a constraint, the relationship between matches is 1-1 (one-to-one), since every transaction will have only two representations separated between two sources, this is our focus on the following chapters. This is also called - constrained matching problem [19][25][24].

2.3 The RL Pipeline

The classical RL pipeline consists in five steps [15]:

- Data pre-processing
- Indexing
- Comparison
- Classification
- Evaluation

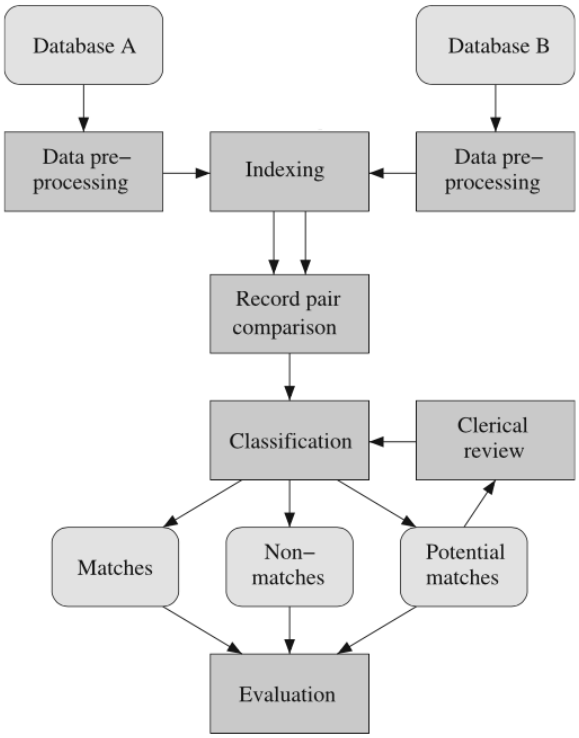

Figure 2.1: Generic RL pipeline as described by Christen in 2012 [15].

The steps and the flow of information can be seen in the figure 2.1, in a simple diagram of a basic non-deterministic RL process. Although this might be the most generic approach, there are many variations to this process. This can be the case, for example, of supervised ML approaches that create a feedback loop in the indexing [5], comparison and classification stages to improve the thresholds used. This allows for each process to be adapted to previously labeled sets of data and it is very useful when training sets are available. The next chapters describe the current state of the art and processes used in each of them.

2.3.1 RL Data Preparation

The ever growing data collection on a variety of areas creates a problem with data consistency. Different systems might represent the same attributes for the same entry in different ways. Other than that, the values of each entry are subject to inconsistencies originated by record entry mistakes and system malfunction [26]. Various levels of data quality across heterogeneous data sources create the necessity for them to be evaluated in terms of how they represent the specific domain in question. This is important in order to the teams configuring the next steps of the RL pipeline to be able to make informed decisions on the right approach to take for each step. Overall, regarding data quality, RL is analogous to data mining processes in the sense that both try to gather meaning from sets of data where it is very difficult to manually evaluate each entry to produce the same outputs. The output result of both system types is directly impacted from the quality of the input data analyzed, [15] which makes preprocessing an intricate part of RL. Regarding the quality, data properties can be divided into [26]:

- Accuracy — inaccurate data can be originated by human error (purposeful or accidental) on value entry, problems with data extraction/gathering mechanisms or technological limitations in the data processing systems.
- Completeness — usually associated with missing values, completeness can be jeopardized by inaccurate records of data modifications, lack of proper record relationship between entries and some attributes can be left out entirely because of the misjudgment of their relevancy.
- Consistency — systems might represent the same attribute with different formats. Values like dates, currency and addresses follow different patterns depending on the standard adopted by the registering system.
- Timeliness — some data domains are time sensitive. When deciding if a portion of a data set maintains its relevance on the study being done, for time sensitive data, the frame chosen can be highly influential. Other common error that affects timeliness is the timing of manual value inputs where the user does not comply with time and date restrictions of the system, when the same restriction rules are not imposed by the same.
- Believably — when there are no truth checks implemented or the origin of the data sets is not confirmed as being trustworthy, there is a question of how believable values are.

While all these data quality properties should be taken into account when data mining, in the RL pipeline, accuracy and consistency are the most influential to the final result.

The difference in data quality in terms defined above is prevalent in real world data and is arguably the reason of the existence of modern RL approaches [15]. Perfect data sets can be linked by deterministic RL like doing a join on a relational database. For all the other cases, in order to produce the best results possible, data needs to be preprossessed. Generically, the result

of this step should be a data set where the values being compared are in the same format and represented by the same data types.

2.3.1.1 Data Cleaning

The data cleaning process aims to reduce incomplete and inconsistent sets of data by smoothing some noise created by anomalies. Each domain might be subject to different types of cleaning in order to achieve the expected results. Some domains might be more sensitive to changes in certain attribute values than others and understanding what this attributes represent to the meaning of each record is key in choosing the right cleaning methods [26]. In RL, other steps of the pipeline are more robust to these anomalies than the standard data mining knowledge discovery and can be more sensitive to smoothing some values. As the process aims to get the best possible match, some data inconsistencies can be overcome by comparison techniques described in chapter 2.3.3. The cleaning of certain attributes of data can create problems while identifying the entity to which several records are related to. The criteria by which certain data cleaning modifications are made should be different for RL than the one applied to data mining models as it should not reduce the ability of the next steps in the pipeline to identify certain entities [15].

Data cleaning steps usually tackle the following data quality issues [26]:

- Missing Values
- Noisy Data
- Value Inconsistency

While handling missing values there are different approaches available depending on the data domain: remove records where missing values are found, remove attributes that have missing values associated with them, manually fill the values, apply the same constant value to all empty entries, fill empty numerical fields with the attribute's median, mean or mode and using rules to fill empty values from attributes that can be easily correlated with other attributes. Not all these options are well suited for most RL systems. Removing records and attributes might result in some records never being matched (mainly in cases where the relationship between records is 1-1), because of either a missing record or attributes that are more relevant to the matching. Filling the values manually, with a constant or a numerical value calculated from other values are dependent on a big knowledge of the domain in question. While manual filling can be feasible for smaller data sets, filling the void with constants or calculated values can be detrimental to RL if entity identification is dependent on specific attribute values. The most common option when dealing with missing values in RL is using rules to decide which values can be automatically filled [15]. For domains where some attributes can be inferred by others, a rule system can be put in place so that the correlation is accurately translated into the missing values. As a simple example, if the date of birth of an individual is known, but the age value is void, the current age can be implied with some certainty by comparing the known value with the current date. The removal process

of inconsistencies in the same record can be related to the rule based option in the empty values case. Easy to correlate attributes can make light of data inconsistencies which can be solved using the same rule approach. Such rules must be implemented only when sufficient knowledge about the weight each attribute have on the entity identification [15]. More on processes to smooth data inconsistencies is described in the next sub-chapter 2.3.1.2. For inconsistencies/noise between different records, there are other options in terms of data cleaning [26]:

- Binning — the process of binning implies an analysis of sorted values and their neighbors. After being sorted, the values are then separated by groups (bins) of a specified length. The values of each group are then replaced by the result of a comparison metric regarding only the groups' values. These metrics are usually the group mean or median. In other cases values in the group can be replaced by its boundaries (maximum and minimum values) depending on which value is closer to the one being replaced.

- Linear Regression — in linear regression, multiple techniques can be applied. The general idea is to find the function of the line which is the best approximation to the distribution of values.

- Outlier Analysis — for the process of outlier analysis, the technique applied is very similar to the one represented in the Indexing step of RL further down this article 2.3.2. Records are aggregated in groups where certain noisy attributes show similarities between them. Then, records that do not fall under any one of the similarity groups are considered outliers and removed from the data set.

Although the three methods above describe part of many data cleaning processes throughout data science, in RL they should not be common practice [15]. The removal of outlier records and smoothing of values might induce the loss of defining characteristics of some records, which might later in the RL pipeline reduce the chance of them being accurately matched.

2.3.1.2 Data Cleaning for RL

The above mentioned data cleaning process might be regarded as a generic solution to data quality problems when preparing data sets to be analyzed. In RL, specially in the Comparison step defined in chapter 2.3.3, one category of data quality metrics plays an important role - consistency across different records [12]. Accurate comparison of record pairs, implies that the values being compared obey to the same standards in terms of how they are represented as much as possible. This is especially true in heterogeneous data sources [15]. The processes about to be described all assume a certain level of knowledge about the domain of the data sets. Although there are multiple combinations of techniques on how to preprocess data for further analysis, in the RL realm, the following are usually used [15][12]:

- String Normalization
- Standardization
- Schema Reconciliation

String Normalization There are many ways how two strings and numbers can represent the same value while being composed by different characters. In order to normalize these values into something comparable, some steps can be taken that while change the written structure of entries, do not interfere with their value. The first and simplest step in string normalization is making all values upper or lower case [12]. This removes the common issue of wrong comparison results when a particular source has strings with uppercase starting characters.

```
# ---------------------------------------------------------------------
# Remove characters and words from input (replace with single space)
             ' ' := '.', '?', '~', '_', ':', ';', '^', '=', ' na ',
                    ' n/a ', ' n.a.', '\', ' also ', ' name ',
                    ' only ', ' abbrev ', ' initials ', ' unk ',
                    ' unkn ', ' missing ', ' unknown '

# Correct words and symbols
         ' and ' := '+', '&'
        ' baby ' := ' babe '
     ' baby of ' := ' babyof ', ' babeof ', ' b/o ', ' b.o.'
    ' known as ' := ' knownas ', ' a.k.a. '

# Remove ' from o'brian etc
            ' o' := " o'"
            ' a' := " a'"
# ---------------------------------------------------------------------
```

Figure 2.2: Lookup table from Febrl [17].

The second step is replacing or removing known unuseful characters and groups of characters. This step implies that a clerical review of the data, by someone well versed in the domain, shows that some data entries have different representations across sources. The usual cases include ordinal number representation, and string separation by dashes or dots. The known cases of sets of tokens that are prevalent in the data sets and can be removed/replaced are manually described in a lookup table like the one in figure 2.2 or by manually defined rules [15]. The process should go through the values and perform modifications if they match one of the patterns in the lookup table or in the rule set.

Standardization The standardization technique can be seen as a more straight forward approach to normalization. Instead of defining rules for very specific cases, the definition relates to known representations of certain data types. Candidates for these types of rules are dates, addresses and

abbreviations. The process, first, follows the same steps as normalization with the creation of a look up table with the desired format for each data type and the possible representations to be converted [12]. If there is enough granularity in the way values are represented, in order to be compared, the process can end here. If some attributes present complex conjunctions of strings, there can be applied a process of tokenization. This last process implies a reference to each part of the target values data type to be described in the look up table. The values are then separated in tokens that comply with the formats defined and added to new attributes defined in the table. An example of this is the separation of a date attribute into day, month and year.

Schema Reconciliation This last technique is only applicable in RL processes working with heterogeneous data sources or when the data produced by the same source was structured differently by the original query. Schema reconciliation is based on the assumption that there is a global schema of the domain in which all sources represent. If source **A** and a source **B** both have different schemas, but they represent the same domain, then there is a possibility of defining two sets of rules M_A and M_B which represent the mapping between schemas of both sources and the original base of knowledge. Such a mapping implies that the original schema is known [44].

2.3.2 Indexing

As stated before, for two groups of data, **A** and **B** with n_a and n_b records each, respectively, the possible matches between all the records included in both groups are the result of $n_a \times n_b$. So, using a naive approach, the possible number of comparisons between the two groups for large data sets is not feasible. Given this possibility, the comparison step in RL is usually the most computationally expensive one in the process.

The objective of the indexing step is to reduce the number of possible comparisons as a way to approximate that value to the one of the possible matches, which in a one-to-one matching scenario would represent the minimum amount of matches between two sources. This step can be related to the filtering of the data. It is a way of creating a more generic similarity measure that the one present in the comparison step. Generically similar records are grouped and only compared between themselves.

The traditional approach to indexing is called blocking [22]. Given a set of attributes all records with the correspondent value will be grouped together. The attributes present in this group are called blocking keys. For clean data sets, this would be enough to get an accurate grouping of records. In scenarios where the data is dirty, there are other methods used. The following chapters will touch on some of those methods.

Although indexing reduces the workload by minimizing the number of false positive matches, it can have drawbacks. In the case of blocking, if two records are placed in different blocks, they will never be compared. This means a bad block allocation might result on the loss of a true positive match between those records. The quality of the filtering applied to the data might result in true matches never being compared, if the criteria for indexing is not well-defined. There are

metrics used both to evaluate the definition of the blocking keys and the quality of the indexing applied [15].

2.3.2.1 Quality Metrics

Choosing the right blocking keys and right index method can have a great impact in the accurate matching of possible connections between records during RL. The blocking keys and the indexing method should be chosen with regard to each other. There are many ways to define blocking keys and indexing methods, even some using available training data to better adjust the choices to the domain in question [5]. With that in mind, one should have a way to characterize the same choices in terms of quality. There are several metrics to consider, but they can generally be divided into the following groups [15]:

- Attribute data quality — the attributes considered to be part of the blocking keys must be complete. While using an indexing technique heavily dependent on a specific attribute, the resulting groups can be affected by said attribute's defects. A field where a lot of records have empty values will not accurately describe them, for example.
- Attribute value frequency — similarly to the previous example, if the distribution of a certain attribute points to a very frequent value, that value will not accurately create a separation between the records it refers to, while using the traditional blocking strategy.
- Number and size of resulting blocks — using very generic blocking keys will result in bigger groups while more specific blocking keys will generate more, smaller, groups. The size of the group can not only impact the amount of comparisons made in the next step. If so, smaller groups would evidently be always the perfect choice as one is trying to reduce the work load of the comparison algorithm. Some indexing methods are sensitive to very specific blocking keys which might lead to wrong group allocations for more ambiguous cases.

2.3.2.2 Blocking

Blocking is the most widespread indexing technique in the field of RL, being one of the first to be introduced [22]. As stated in the definition of the indexing step, to perform the standard blocking, a blocking key is defined by manually selecting a blocking key from a relevant data attribute. Then, all the records with the corresponding values in the same attribute are aggregated in the same group to be compared. Using this method, the frequency and quality of attributes needs to be taken into consideration when choosing the blocking keys [15].

2.3.2.3 Sorted Neighborhood

In sorted neighborhood [28][27], instead of a blocking key, a sorting key is manually selected. The data sources are then sorted according to it and the comparisons generated are done not by creating

dynamic groups of records, following a moving aggregation window. The window size must be fixed and bigger than one (in order to create at least one comparison possible, at a time). Starting at the top of the sorted lists, the window moves through the record and all the possible matches under its domain are compared. A comparison between standard blocking and sorted neighborhood can be seen in figure 2.3 - the shaded cells represent the record comparison candidates to be compared and in the sorted neighborhood, the sorting key is the same as the blocking key in the standard blocking.

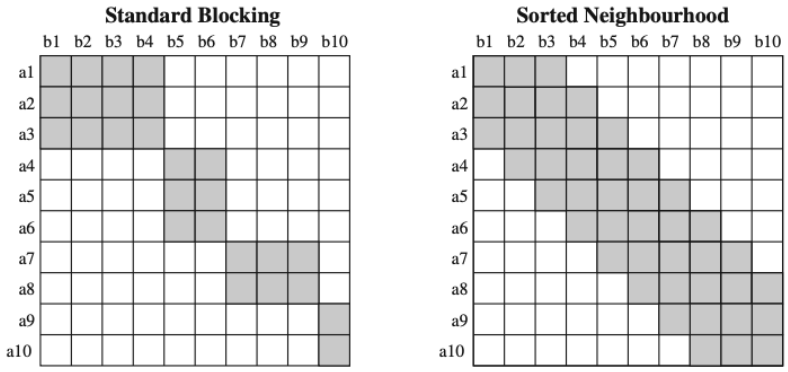

Figure 2.3: Side by side comparison of the standard blocking and sorted neighbourhood indexing methods [15].

The record pairs generated multiple times during the process, are only compared once. So, for each position of the sliding window, there is no repetition of the record pairs to be compared. Unlike the definition of blocking keys, the definition of sorting keys is more sensitive to how the sorting of the data will bring closer similar records.

2.3.2.4 Q-Gram and Suffix-Array

While the previous two indexing methods might work well for clean sets of data, Q-Gram is better suited for dirty data sets [2]. As well as in standard blocking, a blocking key attribute is selected. Instead of creating a different group for records that share the same value for the blocking key, each blocking key is separated into sub strings with the length q (*q-grams*) using a sliding window that goes through the value and add each one to a list. The most common values for q are 2 and 3, or *bigrams* and *trigrams*, respectively [30][50]. Then based on the original list, other lists are created removing one q-gram from the base list. The process repeats iteratively until a list with a minimum length defined by a manually select threshold is found. The record grouping is then done based on the sub lists created by the process, converted into strings. This is much more sensible to variations in the representation of the same data but the process is significantly more expensive than the previous ones.

Similar to q-gram, Suffix array works by generating substrings of the blocking keys [2]. Although, the substrings have to be a suffix of the value. There are two manually defined parameters:

minimum suffix length and maximum block size. Starting with the full string, characters are iteratively eliminated from the beginning until the minimum suffix length is achieved. Each iteration of the string during the process is a possible group. Groups where the number of potential records is bigger than the maximum block size are eliminated. Being similar to q-gram, the drawbacks are also the computational cost of iterating through every value under a certain blocking key. Also, for very large block sizes and small minimum suffix lengths, the resulting comparisons might not compensate the process of indexing using this strategy as many records are likely to have the same two or three letter suffixes. For small block and large minimum suffix sizes, the process might overlook many true match comparisons.

2.3.3 Comparison

Even after data normalization and indexing, the quality of the values of each record is not assured. With that in mind, methods of comparison that rely on a binary classification related to the equality of values do not apply in the majority of cases. Trying to evaluate values from different sources with a 1 and 0 metric might induce in false negative matches between the corresponding records.

R	Attr1	Attr2	Attr3	total
Rec_1	$V1_1$	$V1_2$	$V1_3$	
Rec_2	$V2_1$	$V2_2$	$V2_3$	
Comp	comp($V1_1$,$V2_1$)	comp($V1_2$,$V2_2$)	comp($V1_3$,$V2_3$)	comp_score

Table 2.1: Tabular structure of the comparison vector for two records Rec_1 and Rec_2. The values in Comp represent the individual attribute similarity score between two values. comp_score is the vector total score.

During the comparison step, a vector of similarity is generated as a relative metric for record equality. The concept of similarity comes from the comparison of values present in the records. So, evaluations of true - 1 and false - 0 become: 1- *same*, 0 - *different*. All the values in between are the measure of *how similar* are the two values being compared. This allows for further calculations on how similar two records are in regards to the similarity values of their values. An example of a similarity vector between two records can be seen in table 2.1. The creation of this vector implies all attributes are known and, in the case of comparing two sources, that there is knowledge regarding the corresponding attributes in each one. In cases where some values might be misplaced - the example of names and surnames being switched in some databases - the comparison would return unexpected results and the vector might loose some accuracy. Recently was proposed [52] that attributes with possibly erroneous values switched between them might be placed under an *exchange group*. All the attributes in the exchange group will be compared between them. This increases computation effort in the comparison step but increases the change of accurate matching when facing this constraints. Below are descriptions of comparison methods based on similarity calculation relevant to this study regarding numerical and complex strings.

2.3.3.1 Levenshtein Edit Distance

The edit distance between two string represents the smaller amount of single character additions, deletions and substitutions needed to convert one into the other. Also refered as the Levenshtein distance [40], the method uses a dynamic algorithm to find the optimal edit distance wich represent how dissimilar the values in question are [35].

$$sim_{levenshtein}(s_1, s_2) = 1 - \frac{dist_{levenshtein}(s_1, s_2)}{\max(|s_1|, |s_2|)} \tag{2.1}$$

While calculating similarity for two strings s_1 and s_2 , where $|s_1|$ and $|s_2|$ are the string lengths, the equation 2.1 can be used [15].

2.3.3.2 Numerical

Linking records of financial data makes the case for the need of numerical comparisons with the same metric, similarity, so it can be later combined for the classification process in RL.

As stated for other data types, exact match between two numbers must result in a similarity value of 1. In Christan, 2012 [15], two options are proposed for number similarity measure in RL processes: based on the maximum absolute and relative difference.

When working with absolute values, a maximum difference must be provided, d_{max} ($d_{max} > 0$). The similarity between the two numbers is then calculated relatively to d_{max}. If the absolute difference is bigger than d_{max}, similarity is considered 0.

$$similarity_{abs}(n_1, n_2) = \begin{cases} 1 - (\frac{|n_1 - n_2|}{d_{max}}) & \text{if } |n_1 - n_2| < d_{max} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The relative difference approach, instead of a maximum absolute difference d_{max}, a maximum percentile difference is provided, p_{max} ($0 < p_{max} < 100$). The value used to calculate similarity based on p_{max} is the percentage difference between the two numbers, p_{dif}, equation 2.3.

$$p_{dif} = \frac{|n_1 - n_2|}{\max(|n_1|, |n_2|)} \cdot 100 \tag{2.3}$$

$$similarity_{rel}(n_1, n_2) = \begin{cases} 1 - (\frac{p_{dif}}{p_{max}}) & \text{if } p_{dif} < p_{max} \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

The two formulas for absolute and relative similarity between two numbers n_1 and n_2 are represented in equations 2.2 and 2.4 respectively.

2.3.4 Classification

After indexing and the comparison steps in the RL pipeline, a classification method must be defined in order to attribute a matching evaluation to pairs of records selected to be compared. Traditionally, the variations in the classification methods represent different approaches to how the

individual value comparison groups (comparison vectors, table 2.1) between records are used in order to produce a similarity value. This value then defines in which matching category do the pair fall under.

Some well known examples on how to calculate the final similarity value are [25]:

- Average
- Frequency Based
- Weight Based

The simplest one being the first, calculating the average of comparison values can be used accurately as a scoring technique for clean data sets. The method assumes that every attribute has the same value in terms of how close two records are together. The problem with the average approach is its lack of sensitivity regarding dirty data sets and data sets where some attributes are more relevant than others when comparing records. On the other side, frequency and weight based approaches are more sensitive to the context of each attribute in the domain being evaluated. The main concept behind both approaches is that some attributes are more relevant than others when scoring a similarity vector. If there is a value is very frequent for a determined attribute, usually is not very relevant for records distinction. Giving a higher weight to more heterogeneous attributes is a frequency based decision on how to calculate the final score [51]. Similarly, a clerical review of the data can determine that certain attributes, independent on how frequent, are more valuable to what it means to be a similar record in the context of the domain. That difference is considered a weight based choice on how to calculate the final score.

The attribution of different weigh to different attributes can be done manually if there is sufficient knowledge about the domain. When the combination of that knowledge with consistent data sources allows, the process can result in acceptable results. Although, if either the domain relationships or the data source structure is known, the presence of training sets of already validated data can be used to automatically infer how the weight distribution across the comparison vectors should be done or a combination of this with the automatic choosing of comparison methods for each attribute [4].

Traditional Classification Methods With the resulting data from the calculation of the results of weighted similarity vectors, it is possible to define classifiers that separate candidate record pair into groups: match, not-match and possible match. In the context of this article, two of those methods are worth mentioning - probabilistic and rule-based classification.

2.3.4.1 Probabilistic

Probabilistic classification has been a staple in record linkage since its introduction in 1959 [41] and formalization in 1969 [22] until today.

The probabilistic approach takes into consideration the results from the comparison vectors, only using binary evaluations for each value comparison, and separates pairs of records into non-matches, possible matches and matches. The resulting comparison vector score is then compared to two fixed thresholds. Score values above the higher threshold fall under the matching group and below the minimum threshold fall under non-match. Every score in between falls under possible match. Record pairs under possible match are then assumed to be evaluated by clerical review [25].

In probabilistic record linkage the evaluation methods used and threshold values should be tuned to the domain of the data set in question.

2.3.4.2 Rule Based

The rule based approach applies rules to pre-determined values of comparison for select attributes. After attribute selection, rules are then defined for each result in the comparison vector. Opposite to the traditional probabilistic approach, rules can be applied to comparison values between 0 and 1, instead of binary evaluation paradigm. Multiple rules can be defined. The rules are then combined by logical operators: negation, conjunction and disjunction [18] [39].

After the creation of the testing predicate, composed by the rules and operators, record pair comparison vectors are then evaluated. If the predicate returns a true value, record pairs are selected as a match, otherwise, they are selected as non-match. Rule based classifiers are a good approach for cases where the domain is very well known by the individual creating the predicates. Another use case where the use of rule based classifiers can be successfully applied is when there is already some test data with relevant matching information about record pairs from the data sources domain. Test data enables the opportunity to test the rules for coverage, which is an accurate indicator of how well broad is the rules' inference power over the data sets and nuances in the data [15].

Chapter 3

Supervised Machine Learning for RL

In the previous chapter, there was an introduction to how the RL process works. All the methods presented are based on possible combinations of data where one looks to match records with the resource of statistical analysis and knowledge of the record relations in question. The lack of previously evaluated data in regard to matching relationship, creates the necessity for the above mentioned methods. This methodology, unsupervised, is well suited for clean data sets with very distinctive attributes where the notion of what defines a specific entity in the context of the domain is clear for the user defining the rules, comparison functions and fixed threshold values. For data sets with less favorable combinations of characteristics, the process yields less desirable results [31].

For target data sets where analogous sets of previously verified matches are available, there is the possibility of creating evaluation models to help in different steps of the RL pipeline [15].

3.1 Generic model generation for RL Classification

It has been proposed [21] that, for the RL classification problem, the training set for a supervised classifier should be created as group of instances of the similarity vectors between records of the same domain and the corresponding matching status. In supervised classification, the model is created based on a set of features and labels for each record in the training data. In the case of RL test data sets, the records are the similarity vectors created using the same similarity methods as the ones in the comparison step of the RL pipeline. The label of each record is the matching status of the record pair that originated the similarity vector and the features correspond to the similarity values of each pair of values.

To prevent over-fitting to a specific set of training data [37], the model can then be evaluated by testing its ability to accurately classify records from a different set of training data. In order to evaluate the accuracy of the classifier, there are several techniques with different takes on the division of training data that can be used [33]. The most common one, *hold-out*, is achieved by dividing the training set into n folds. While two parts are used to generate a single model, the third is ran through the model as a means to test the performance. The second method is called

cross-validation. To achieve this, the training data is separated into a specified number of groups of equal size. The model is then iterative trained with all groups, except for one, and tested for the group that was left behind. The performance of the classifier can be calculated by the average of all the iterations of the model generated. The last method is an extreme variation of *cross-validation* called *leave-one-out*. In this last method, the groups are composed of a single record of the training data. The rest of the process is the same. Although computationally much more expensive, *leave-one-out* can produce much more accurate performance estimations.

Other considerations when generating models for RL classification include the match and non-match classification imbalance in the training sets. Usually, when very large sets are available, the percentage of true matches is significantly smaller than non-matches. This imbalance can take a toll on model performance [1]. A possible solution to this was proposed [29], where synthetic training data is created based on the features of the real training set.

3.2 Examples of Binary Classification Algorithms in RL

Multiple implementations of RL frameworks using supervised classification have been implemented across different data domains. Examples include: Active Atlas [48][49], MARLIN [6][7], Multiple Classifier System [53], TAILOR [21], FEBRL [16][17], STEM [31] and Context Based Framework [14].

Among these, the some frequently used algorithms for supervised binary classification are decision trees, logistic regression, Naive Bayes, support vector machines (SVMs) [32] and algorithm boosting techniques such as AdaBoost and Random Forest. Both SVMs and decision trees are the most popular choice among RL implementations [15] and yield the best results [1].

3.2.1 Logistic Regression

The logistic regression is a popular binary classification method. The algorithm is based on a function called the sigmoid function, or logistic function, described by the equation 3.1. The algorithm maps the values to an interval between 0 and 1, using 0.5 as the threshold for the binary classification.

$$f(x) = \frac{1}{1+e^{-}(x)} \tag{3.1}$$

Logistic regression performs better with evident linear relationship between the features and labels, but can be very sensitive to highly correlated inputs.

3.2.2 Naive Bayes

The Naive Bayes classification algorithm is based in the Bayes' theorem, represented in equation 3.2 where C and F represent possible classes and independent variables of each entry, respectively

[34].

$$P(C|F) = \frac{P(F|C) \cdot P(C)}{P(F)} \tag{3.2}$$

The overall performance of Naive Bayes can be affected by the underlying assumption of variable independence.

3.2.3 Decision trees

Decision Trees fall under the category of **logic based algorithms** for supervised classification. Generically they are used as a means to represent rules that describe a hierarchical division of a data set. This means decision tree can be used in other data science tasks other than classification such as data description and generalization [38]. In this article the focus will be on the classification applications of this supervised method. Decision trees classify instances by sorting their features, starting at the root, and following the branches that represent the evaluation regarding the value of the feature being considered. The nodes represent features while the branches represent the possible values the feature can take [33]. Several methods can be applied in order to construct a decision tree. The common denominator is that the root node should be the feature that best divides the data set into the two possible classes [38]. The final result, is a set of rules regarding the features values, ordered (starting at the root node) from the most divisive to the least. In the case of RL the first value evaluated, when the tree is empty, corresponds to the one which can better divide the entire set into one of the two possible outcomes: match or non-match. Then, leaves are generated by choosing the values that better match the splitting criteria defined. The process can be related to the rule based classification described in the previous chapter. The final result is a predicate composed by a set of rules that maximize the segregation of record pairs depending on the most controversial attributes [15]. The fact that decision trees are easy to visualize and can be related to a rule based approach, makes them an easy to understand [33].

3.2.4 Support Vector Machines

The concept behind Support Vector Machines (SVMs) involves the maximization of the distance between data instances of opposing classes, separated by a hyperplane. This can be easily achieved in linear separable classes, but it is not the case for most real world data. In cases where instances of different classes are not separable linearly, the data must then be plotted into a multi dimensional space (with the maximum number of dimensions possible being the number of features). This can be achieved by using a *kernel* function, selected based on the type of data distribution, that can map the feature values into the multi dimensional space. The function of the multi dimensional boundary (hyperplane) is then defined by the maximum possible distance to the closest instances of different classes, called *support vectors*. When classifying new instances, the same *kernel* function used in the model creation is then used to plot them in the multi dimensional space. The classification is based on which side of the hyperspace the instance is plotted into [11]. There is no standard *kernel* function to apply in the majority of use cases of SVMs. It is accepted that a

range of *kernel* function settings can be considered and then tested to evaluate which one adapts better to the problem and produces the best separation and consequent classification [33].

A recent survey [1] showed the clear advantages in terms of matching performance of SVMs and decision trees as classification systems in RL over other supervised and unsupervised methods. Although SVMs produce the best results for true matching percentage, the amount of training data needed can make the process not applicable for most cases. SVM algorithms need more fine-tuning than decision trees, given the complexity of the process of mapping feature vectors into the multi dimensional space by selecting *kernel* settings. The same complexity applies to the calculation of multi dimentional vector distances which requires high computational power and makes the SVM model generation one of the slowest among supervised classification algorithms.

3.2.5 Ensemble Methods

There have been many proposals for a predictor aggregation technique that can outperform a single algorithm by combining multiple iterations of the same or different predictors. Between this, two widespread techniques with well known implementations based on decision trees are: boosting [46] and bagging [9]. The main difference between the two approaches lies in the relationship of different iterations of the predictors generated. In bagging, bootstrap sets of the training set are randomly selected and used in order to generate a number of independent predictors. In the case of boosting, the entire training data set is used in each step and the new iteration of predictors is dependent on the previous ones by trying to fix the shortcomings inherited by previous generations [34]. In both cases, the final predicting performance is a combination of all the predictors generated. In the case of bagging, the main outcome is a decrease in bias caused by the introduction of a random set of information in each training while in boosting, the main advantage of the final prediction is a reduction of the misclassification error rate. The two main boosting and bagging systems using decision trees are AdaBoost [23] and Random Forests [10], respectively.

3.3 Model Evaluation Metrics

Classification tasks' performance can be evaluated by very reductive metrics regarding the outcome of the predictors. The choice of the right metric is the first step in order to understand the value of the choices made while choosing and configuring a classification algorithm.

		Real	
		Positive	**Negative**
Predicted	**Positive**	TP	FP
	Negative	FN	TN

Table 3.1: Confusion matrix. TP - true positive, FP - false positive, FN - false negative, TN - true negative.

Binary classification test results (like in record linkage classification) can be presented in a tabular form known as a confusion matrix (table 3.1). All the elementary metrics for binary classification performance can be derived from the confusion matrix.

Metric	**Function**
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
Error Rate	$\frac{FP+FN}{TP+FP+FN+TN}$
Recall	$\frac{TP}{TP+FN}$
Specificity	$\frac{TN}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
Negative Predictive Value	$\frac{TN}{FN+TN}$
False Discovery Rate	$\frac{FP}{FP+TP}$
False Positive Rate	$\frac{FP}{FP+TN}$

Table 3.2: Elementary performance measures as defined in [3].

Given the nature of the indexing step in record linkage, the training and testing data sets can present a heavy imbalance towards the negative class. In order to make an informed decision, the metrics defined in table 3.2 can be combined into composite performance metrics that are robust enough to be transparent regarding that imbalance or can be tuned to reflect that imbalance.

Usually, for well balanced datasets, a popular combined metric is the area under the *receiving operating characteristics* curve (AUC ROC). The ROC represents the balance between the recall and the false positive rate, but can be too optimistic when used to evaluate models trained and tested with an imbalanced dataset. On the other hand, the AUC of the *precision-recall* curve is strongly related to the AUC ROC, while being more accurate when dealing with imbalance in the dataset [8]. Two good combined metrics that have proven more informative for the same datasets are F_β, equation 3.3 and the Matthews Correlation Coefficient, eq. 3.4 [47].

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{3.3}$$

The F_β score represents the harmonic mean of the precision and recall for a β value of 1. For imbalance towards the negative values, lower values of β allow for a more useful metric of

performance.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FN)(TP+FP)(TN+FP)(TN+FN)}} \tag{3.4}$$

The Matthews Correlation Coefficient score is a value between -1 and 1 where the extremes represent perfect negative correlation (all negatives as positives, all positives as negatives) and perfect positive correlation (all predictions are correct), respectively. When the value is zero, the model is the equivalent of a random classifier.

3.4 Other Applications

The availability of validated training sets is not only useful to create models for the classification step in RL. Though the pipeline, many choices have to be made by manual evaluation of the data set domain. Is not always evident what is the right combination of tools that provides the best matching results. This is the reason why so many frameworks exist in the field of RL [32], there is no single solution to all the possible applications.

Some supervised methods have been proposed try to replace some manual interaction with the RL system. The choosing of the best blocking method for the indexing step based on training sets of rules for blocks of records is a good example of how this can be achieved [5][36]. Although, most notably, a more general approach was proposed by Köpcke and Rahm [31], STEM. The proposal is a framework that combines the use of multiple supervised algorithms to choose the best combination of elements in the RL pipeline for the data sets in question.

Chapter 4

Reconciliations at Natixis CIB Department

Every day Natixis' traders perform transactions in every open market in the world. A transaction is manually registered in an internal platform and is then replicated on the external platform corresponding to the type commodity being transferred in the market in question. In order to keep track of the traders' portfolio, the bank needs to be assured that the mutual agreement, made with the third party, corresponds to the information stored in the internal system.

Since the internal process of registering a transaction is manual and the external registering process is unknown to the bank, it is error prone. External providers often send their transaction data to be cross analyzed by Natixis' employees and the bank has the opportunity to make amendments to transaction data in case any discrepancies are detected.

The anti-fraud team is responsible for the verification of the transaction data for the whole bank, for every commodity, in every market. The verification should happen as fast as possible.

4.1 Data Sources and Quality

Transactions are not isolated incidents. What is commonly known as a transaction, represents a stream of different operations, financial flows. When a trader initiates a new flow, its characteristics are communicated to the anti-fraud team. New flows are known as new reconciliations. Reconciliations characteristics are:

- Market geolocation
- External Platform Identification
- Internal Platform Identification
- Type of asset
- Currency

- Expiration Date (in some cases)

At the same time, both providers start sending data regarding transactions in the new reconciliation. The data generated in each platform is the result of a query that is system specific and unknown to the team. All the query results are sent in CSV format and placed in a secure network shared location. Because Natixis uses different trading platforms internally for different types of transactions and registers them in different external platforms depending on the commodities and markets, there is no standard format for how the data is structured.

Platforms produce data in the form of TXT and CSV files. The only common feature is that different lines represent different transactions. In the same line, values are either ',' or ';' separated. While some providers send one file per transaction, in tabular format, with attribute description, others omit the attribute descriptions or even send multiple reconciliations per file with no particular order. In terms of how the files are created, in some cases, every day a new file is generated with the transactions processed by the time of creation while others send all the transactions generated from the beginning of the year alongside with the new ones.

In summary, regarding data quality sources present the following defects:

- Absence of a standard schema — sources do not gather information from databases with the same schema organization. At the same time, for the same source, different queries are applied in order to obtain information of different reconciliations.

- No standard formats — since sources are platforms used in different parts of the world, date, currency and numerical values are not represented in the same way.

- Missing values — when deemed not useful, some sources might omit values or even entire attributes from the data sets provided.

4.2 Current System

The current reconciliation system is composed by three modules: data processor, matcher and validator. When new data is received by the platforms, is normalized, cleaned and saved to an SQL database. The records are then matched in pairs (external and internal representations of the same transaction) and displayed in a dashboard to be compared manually by a user that checks for discrepancies, missing values or evaluates problems with the matching when they occur. An approximation of this system can be seen in the figure 2.1.

When a new reconciliation is inserted into the system, a business analyst goes through the unmatched transactions and gives inputs regarding two sets of properties:

- Relevant sets of values for the transaction — platforms include irrelevant data for the reconciliation process, for example the name of the person who registered it into the system and the time of registering.

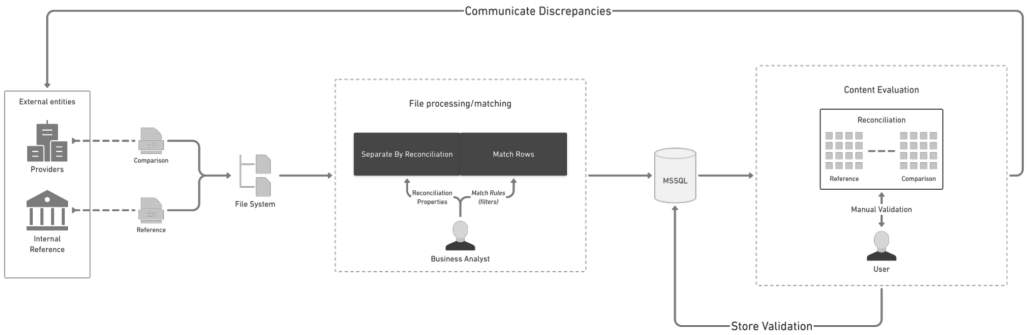

Figure 4.1: Representation of the current Natixis reconciliation system.

- Casuistic data normalization — representations of currency, date formats, floating points and string modifications such as the exclusion of prefixes and suffixes.

These inputs are communicated by the business analyst to the developer team which implements them by creating sets of rules which are then saved to a table in the SQL database. Rules can be in one of two categories:

- Match Rules — rules used to match corresponding transactions across two systems.
- Gap Rules — rules used to validate the consistency of the data present in the two systems.

This is a trial and error approach as sometimes the business analyst inferences can be wrong and some transactions are unable to be matched with success. At the same time, platforms occasionally make small changes to the queries and the rules become obsolete and have to be rechecked.

Chapter 5

Methodology

The main purpose of this study is to evaluate the feasibility of implementing a record linkage system, backed up by supervised machine learning, in order to reduce an redirect efforts of the reconciliation team at Natixis in the day-to-day task of reconciling the banks' transaction information across several markets.

The heterogeneous nature of the data handled by the team and occasional small changes to the sources' system to output the data corresponding to the transactions to be reconciled, make the scope of the problem very broad. From the start of the process, where raw CSV files are ingested by the main system, to the end goal, matching the corresponding transactions and validating data is consistent across both representations, the lines that separate the objective of each step can be blurry. A good example of this are *matching* and *gap* rules which both play a key role in two different steps of the process - matching transactions and validating the data present in both, respectively - but can be used at the same time to represent relationships between sources, merging the two steps.

Being a mission critical application inside the bank, the reconciliation system does not have much room for radical changes across the different steps of the process. Proposals for improvement are thoroughly vetted by multiple parties that focus in different aspects of its implementation, roll out to production and outcomes. It has become evident, after multiple discussions with personnel involved, that any proposal for improvement must include considerations for:

- Modularity — ease of partial integration in the current system, for testing, while maintaining legacy fallback modules.
- Budget Sensitivity — fall between the team's budget for computational power and developers' work hours. This applies for implementation and long term maintenance.
- Explainability — the concepts should be relatively easy to transmit to stakeholders.

Taking all this into consideration, the scope of the proposed solution by this study was narrowed down to the phase after the data is parsed from CSV files, semi-normalized, separated by reconciliation and saved to an SQL database. Each reconciliation will be treated as a separate

problem, with its own record linkage pipeline set of parameters, including a classification model. Instead of matching transactions and subsequently validating the values, only valid matches are to be signaled as *matches*. This approach maintains the teams' structure where the business analysts and users work in specific reconciliations, independent from others, and aims to compare the robustness of a supervised record linkage system in comparison to the current set of rules. All the steps are thought as to be inside the constraints defined by the bank while trying to maximize the end results.

Currently, inside Natixis, other efforts have been done in order to semi-automate the reconciliation system. These proposals are centered in matching different data attributes across providers using supervised machine learning. In the future, the outcomes can be used as input for the record linkage pipeline.

5.1 Data Gathering

In order to train and evaluate the system, there are two groups of reconciliation data to gather: transaction data and rules. All data was extracted from a backup production server containing active and inactive reconciliation information, since the inception of the project. The lack of an API for data extraction or a clear database documentation made this a task of having to learn the database schema and generating RAW SQL queries to access the data in question. The database schema knowledge was attained though multiple meetings with the team leader of the project.

5.1.1 Transaction Data

After being processed by the first phase of the system, CSV data is separated into provider data tables. Since there is no information regarding the direct column correspondence, there is no standard tabular scheme for all providers. Each provider is represented by a separate table where the column names are the same as represented in the origin file. Data types are only identified and separated in three possible groups - numbers, dates and strings. Natixis' privacy policy allows for a very narrow and controlled access to the data in question. Given the lack of a proper development machine for data extraction and preparation, the strategy passes through a process of getting the raw values and processing them in a system outside the bank's infrastructure. This allowed for the extraction of the historic match results of one reconciliation, across its entire lifetime. This includes the transaction data and the mapping for *match* records between the two sources.

5.1.2 Rules

In order to create some type of baseline for the process in terms of ability to match transactions, there is a need to extract some previously selected relationship between the attributes of each provider table. This relationship is translated in the *gap* and *match* rules. Rules are written in Transact SQL and are saved as strings, in the database. Rules can be as simple as a 1:1 relationship

between attributes, or as complex as multiple combinations of attributes with numeric operations or string concatenations.

5.2 Data Cleaning

The transaction data and rules present in the system have a low level of parsing. The main data cleaning task performed is related to the transaction values. The CSV files are often in non-tabular format which are then translated though a chain of rules to generate the provider table. Other than that, the only normalization task performed is the formatting of the date type arguments. Rules are subject to less data cleaning tasks and are represented, as described in the sub-section 5.1.2, by a single string containing a Transact SQL valid statement.

5.2.1 Transaction Data

Usually, the data cleaning task would take into consideration the context of each attribute and the data quality of the sample. In this case, the sample is pre-validated, so the data is noiseless. There is, although, an issue with missing attributes. This is due to the fact that, for the same provider, not every reconciliation is represented by the same or all the attributes in the provider table. A cross validation between the recorded reconciliation parameters, from when it was first introduced in the system, allowed for the extraction of only the relevant parameters from the provider table. Relative to normalization, the only step necessary to achieve accurate results in next steps, was to remove special characters from strings and set every value to lowercase [12] [15].

5.2.2 Rules

In the context of data cleaning for this process, the rules are a bigger challenge than the transaction data. The fact that there is no parser available fully capable of decomposing the Transact SQL statements into the different column names of each reference and comparison source tables, along with the complexity of existent rules, was mitigated by using a mixed approach of lookup tables and regular expressions. The attribute names involved were extracted, with disregard for a lot of the relationship information.

The scope of the solution is based on a naive representation of the relationship between attributes, so rules containing statements with combinations of attributes are dropped. Similarly, statements containing functions applied to an attribute, or relationships other than direct correspondence, are translated into a, simple, direct correspondence - attribute-X-A:attribute-Z-B, for two providers **A** and **B** containing attributes **X** and **Z**, respectively. An example of this process is represented in the table 5.1 The rules were then de-duplicated to remove redundancies.

One of the main issues with this step, was the data quality. Not only some non valid Transact SQL statements were present, making the data inconsistent, but there were other miscellaneous problems that increased the level of difficulty of this step such as - missing values, inconsistent naming schemes for attributes, attributes prefixed with the wrong source (*comp* for comparison

Input	Output
CONVERT(VARCHAR,Ref.COLNAME_R,103) = CONVERT(VARCHAR,Comp.COLNAME_C,103)	[COLNAME_R, COLNAME_C]
CONVERT(DECIMAL(28,6), Ref.COLNAME_R) = (WHEN Comp.COLNAME_C <>'XYZ' THEN CONVERT(DECIMAL(28, 6), ABS(Comp.COLNAME_C)) ELSE CONVERT(DECIMAL(28, 6), Comp.COLNAME_C)	[COLNAME_R, COLNAME_C]

Table 5.1: Rule parsing example. Real data values are ofuscated for security reasons. COLNAME_R and COLNAME_C correspond to names of the attributes in the reference and comparison sources, respectively.

or *ref* for reference) and typos. The final result is a naive representation of the rule set, for a reconciliation, but acts as a good baseline for the robustness of a record linkage pipeline in this context, with a fraction of the total information in the current system, while at the same time creating the bare minimum relationship set.

5.3 Indexing

While approaching the indexing step, the goal is to create the least amount of comparison samples possible. With this in mind, a blocking technique was applied. To perform this operation, a custom version of the Python Record Linkage Toolkit [20] was used. The standard toolkit uses a Pandas [42] standard comparison algorithm to create the comparison blocks, but in order to gain flexibility, the comparison algorithm used was the Levenshtein Edit Distance [35]. This allowed for the definition of a threshold for creating the least amount of blocks while maintaining all the matched pairs in the same block.

5.4 Comparison

In order to generate the comparison vectors to train the classification models, the transactions of each indexing group were compared. The combination of attributes to compare between indexed transactions was based on the output of the rule cleaning step, section 5.2.2. The result, in table 5.2, was a matrix where each row represents a comparison between a transaction in the reference and comparison systems. The attributes represent a rule, a comparison of the two columns described by the rule on each system. The entries take any value between 0 and 1 corresponding, respectively, to *completely different* and *exactly the same*.

Since no knowledge about attribute meaning in each source was taken into account, the comparison algorithms used for every attribute was based exclusively on the data type: Levenshtein Edit Distance for strings and a linear numeric comparison with no scaling or offset. Using the mapping for each matched pair retrieved with the transaction data, a new attribute was added to the matrix with a binary label - 0 for non-match and 1 for match.

ref_id	comp_id	rule_1	rule_2	rule_3	rule_4	...	rule_n	match
12345	98765	0.3	1	0.23	0.1	...	0.6	1
19765	67834	0.6	0.2	0.9	0.35	...	0.4	0
...	...	...	...	...	...	...	...	...
65483	90876	0.7	0.95	0.17	0.2		0.65	1

Table 5.2: Comparison output example. All data present in the table is synthetic for security reasons. ref and comp id represent the transaction id in reference and comparison sources, respectively. The attributes represent the similarity result between the columns joined by each rule.

5.5 Tuning

Before the classification phase, an analysis of the comparison matrix was performed to remove any rules resulting in a 0 similarity score for every value. This was also useful to evaluate the thresholds for the indexing step while maintaining 100% of the matching transactions in the same group.

5.6 Classification

The problem at this stage was considered as a binary classification task. All the algorithms and metrics calculations described in the next steps were the scikit-learn [43] implementation of the same methods.

5.6.1 Evaluation Metrics

One of the particularities of the record linkage classification step is its sensitivity in terms of class imbalance to the strategy adopted in the indexing step. In most cases, mainly due to the simplistic rule set used, the *non-match* entries in the training data will heavily outweigh the *matches*. The metrics used needed to reflect an accurate description of the classification model's performance without disregard for this imbalance. For an overall overview of the performance, the AUC of the Precision/Recall curve and the Matthews correlation coefficient were used. For the bank, the correct classification of *non-matches* is more relevant. For that reason the F_β score with a β value of 0.5 was used.

5.6.2 Classification Algorithms

To create a baseline for future comparison, a naive random classifier and some classification models were used. The classification algorithms used were Naive Bayes, logistic regression and, for ensemble algorithms, AdaBoost and Random Forest. The AdaBoost and Random Forest algorithms were based in CART decision trees and the Naive Bayes implementation is based on the proposal by Rennie et al. [45], described in sklearn as Complement Naive Bayes. The remaining hyperparameters were set as default values from the sklearn library and performance metrics were

calculated from the average of all models using cross-validation with a fold number of 5. The algorithms were chosen based on the interpretability factor, previous usage in machine learning applications inside the bank and the fact that the current data science team working in the reconciliation space is familiar with boosting and Random Forest. From all the tested algorithms is possible not only obtain the final classification as well as the list of probabilities of classifying each sample. The training data was a sample with 5k matches randomly selected from the reconciliation's lifetime.

5.6.3 Hyperparameter Tuning

Given the promising results and the fact that can be parallelized for faster fitting and predicting (a good match for the budget sensitivity of the task), the Random Forest algorithm was selected to be tuned. In order to proceed with the Hyperparameter tuning, instead of a computationally expensive run of Grid Search inside the bank's development machines, the process used was the Google Vizier implementation in the Google Cloud AI platform. The hyperparameter optimization algorithm chosen was the Bayesian Optimization with the objective of maximizing the F_β score, with $\beta = 0.5$, of the model. All the runs used cross-validation with a fold number of 5. The parameters evaluated and the correspondent value interval were: maximum depth of each tree - [10, 30], the minimum amount of samples to consider for a split (percentage of total samples) - [0.001, 0.1], the number of features to consider in each split (percentage of total) - [0.1, 1] and the number of samples to use in the bootstrapped data set (percentage of total) - [0.1, 0.9]. The value for the number of trees was set to 1k.

5.6.4 Feature Importance and Selection

In order to better understand the weight of each rule being considered in the final classification, as well as to try to improve the final model's performance, the Random Forest model *feature importance* output was considered. This methodology was selected as Random Forest was showing the best results and already includes a feature selection method. The model was trained iteratively by removing the rule with the lower importance value from the last iteration. This was done until the metrics for the model performance evaluation started to show a downward trend. In order to increase feature importance evidence, the model from which the feature importance list was extracted, was generated using 5k trees.

Chapter 6

Results and Discussion

As stated in the Methodology, chapter 5, the data set used was a random sample of 5k transaction matches from the same reconciliation.

6.1 Indexing and Comparison

After cleaning the data, removing duplicates and separating reference from comparison data, the total of 182 rules in the system were reduced to 19. The transaction data analysis showed that for each reference transaction, there are one or more comparison transactions. For 5k matches, the reference unique transaction count is 5k while the comparison set represented only 2660 unique transactions.

	Indexing Technique	
	Full	**Blocking**
Total Comparisons	13300000	68305
Comparison Time (seconds)	2410	13
Match/Non-Match Ratio	0.0004	0.0732

Table 6.1: Total comparisons by indexing technique. The Match/Non-Match ratio represents the class imbalance.

After applying a full index technique, the amount of comparisons made and resultant comparison vectors were 13300000. The process takes around 2420 seconds. After analyzing the comparison results, in table 6.2, the rule with index 12 was selected as value for blocking since every *match* presented a value of 1 for that specific comparison. The process was repeated which resulted in a total of 68305 comparisons indexed with no loss of matches. The comparison step took 13 seconds as seen in table 6.1.

	mean	std. deviation	min	max
rule_0	0.2328	0.4222	0.0	1.0
rule_1	0.2844	0.4508	0.0	1.0
rule_2	0.4388	0.4309	0.0	1.0
rule_3	0.596	0.4148	0.0	1.0
rule_4	0.802	0.2142	0.4	1.0
rule_5	0.6686	0.4589	0.0	1.0
rule_6	0.4497	0.4974	0.0	1.0
rule_7	0.5316	0.4989	0.0	1.0
rule_8	0.6686	0.4589	0.0	1.0
rule_9	0.2794	0.4482	0.0	1.0
rule_10	0.2294	0.4196	0.0	1.0
rule_11	0.5937	0.4147	0.0	1.0
rule_12	1.0	0.0	1.0	1.0
rule_13	0.7925	0.2228	0.1579	1.0
rule_14	0.4388	0.4309	0.0	1.0
rule_15	0.5262	0.4987	0.0	1.0
rule_16	0.4436	0.4965	0.0	1.0
rule_17	0.3865	0.3713	0.0	0.75
rule_18	0.3846	0.3709	0.0	0.75

Table 6.2: Analysis of the comparison vectors for *matched* transactions present in the comparison results.

6.2 Classification

6.2.1 Baseline

Algorithm	F0.5	M. Coef.	AUC PR	AUC ROC	Running Time (seconds)
Random Forest	0.9127	0.9334	0.9829	0.9992	44.69
AdaBoost	0.8749	0.9023	0.9710	0.9987	2.42
Logistic Regression	0.8364	0.8575	0.9445	0.9975	3.18
Naive Bayes	0.6544	0.7590	0.8749	0.9953	0.08

Table 6.3: Comparison of model metrics between the algorithms tested without tuning. F0.5 is the result of F_*beta* with *_beta* = 0.5, M. Coef. is the Matthews Coefficient, AUC PR is the area under the precision/recall curve and AUC ROC is the area under the ROC curve.

To create a baseline for the classification performance, Naive Bayes, logistic regression, AdaBoost and Random Forest were used. The results are compared based on the metrics defined in section 5.6.1 and can be seen in table 6.3.

Here became evident that AUC ROC is not a good metric for a imbalanced binary classification problems. Random Forest was the overall best performer and was chosen to be tuned. The results are also demonstrated in figure 6.1. Random Forest has proven to be more consistent across both precision and recall values for each prediction.

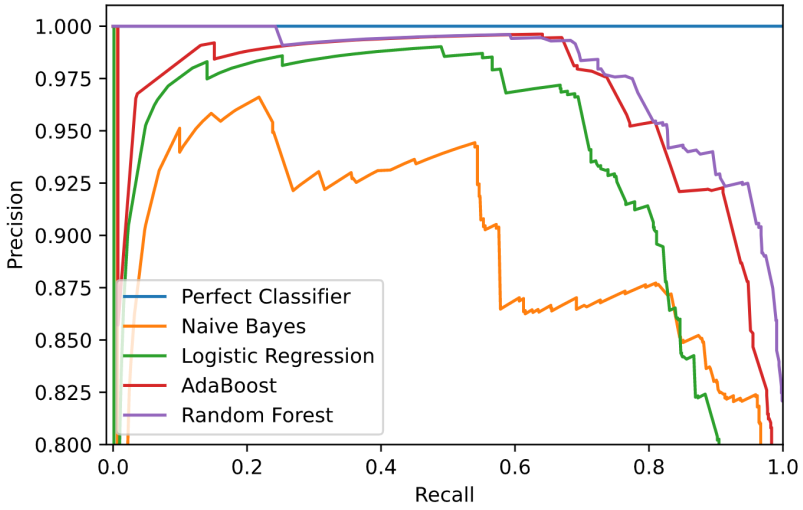

Figure 6.1: Precision/Recall curve of each algorithm tested.

6.2.2 Random Forest Classification

6.2.2.1 Tuning

The run of hyperparameter tuning in Google Cloud Ai, with 200 iterations, returned the values present in table 6.4 as the ones which maximize the F_β score, with $\beta = 0.5$.

Random Forest Tuned Parameters	
max-depth	27
min-samples-split	0.0009
max-features	0.37
max-samples	0.9

Table 6.4: Google Cloud Bayesian hyperparameter tuning of the Random Forest algorithm outputs after 200 iterations.

After training the model with the tuned values, the results had a slight increase as shown in F0.5 and Mathews Coefficient as seen in table 6.5. The AUC of the Precision/Recall curve did not show any significant improvement due to the fact that, even though the amount of *non-match* correctly predicted increased, the *match* prediction error increased. This change can be seen in the normalized confusion matrices, for the model before (figure 6.2) and after (figure 6.3).

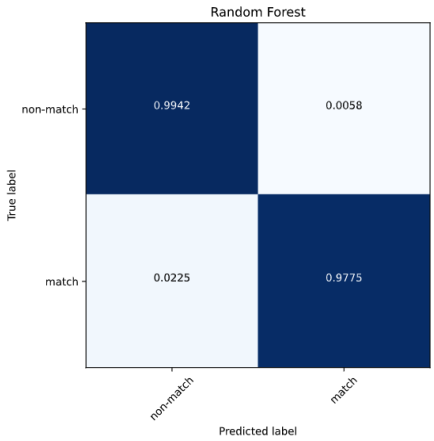

Figure 6.2: Normalized confusion matrix for the model before the hyperparameter tuning.

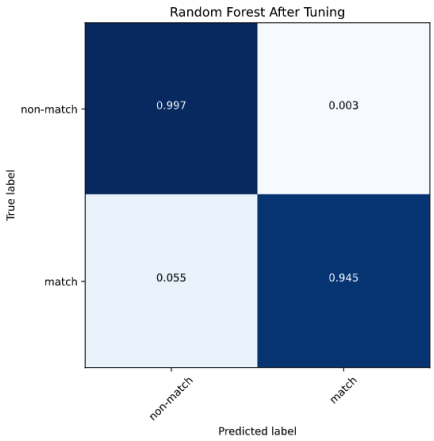

Figure 6.3: Normalized confusion matrix for the model after the hyperparameter tuning.

Algorithm	F0.5	M. Coef.	AUC PR	AUC ROC	Running Time (secs)
Random Forest	0.9127	0.9334	0.9829	0.9992	44.69
Random Forest Tuned	0.9437	0.9413	0.9894	0.9988	24.94

Table 6.5: Comparison of model metrics between Random Forest before and after Google Cloud Hyperparameter Tuning. F0.5 is the result of F_$beta$ with _$beta = 0.5$, M. Coef. is the Matthews Coefficient, AUC PR is the area under the precision/recall curve and AUC ROC is the area under the ROC curve.

6.2.2.2 Feature Importance and Selection

After tuning the parameters, a new model was generated using 5k trees as the number of estimators. From this model, a set of feature importance was extracted and can be seen, with features sorted by order of importance, in figure 6.4. Each feature is represented by the index of the rule which is in its origin.

The model was then trained iteratively by removing the feature with the lower value of importance from the previous iteration feature set. The test was stopped once the performance metrics started to score lower than the baseline defined in section 6.2.2.1. The best outcome was achieved by removing the least important feature, *rule_8*, using 18 features in total, as it is presented in table 6.6.

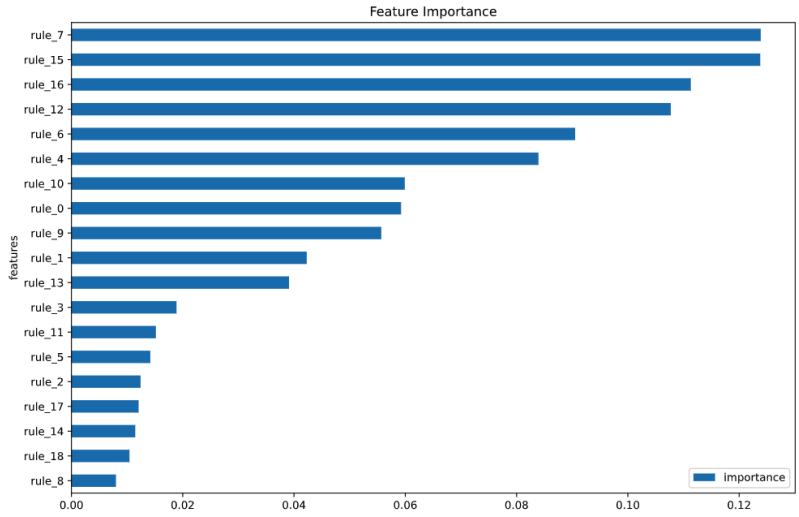

Figure 6.4: Feature importance values calculated using a Random Forest algorithm with 5k predictor trees.

Features used	F0.5	M. Coef.	AUC PR	AUC ROC	Running Time (seconds)
19	0.9437	0.9413	0.9894	0.9988	24.94
18	0.9446	0.9428	0.9896	0.9989	22.41
17	0.9441	0.9414	0.9896	0.9989	22.22
16	0.9440	0.9425	0.9895	0.9988	24.01
15	0.9440	0.9415	0.9890	0.9989	22.52

Table 6.6: Results of the iterations over the feature importance values. Each new iteration was done by training the model removing the least important feature in the precious one.

6.3 Remarks

In order for a record linkage pipeline to work as a replacement for the current system, there must be an indication of robustness for lack of information about reconciliations while at the same time have room for fine tuning by the business analysts.

The current approach shows promising results as from a naive standpoint, regarding how the sources attributes are related to each other. For a single reconciliation, with around 10% of the total rules agragetted overtime by analysts, it is possible to predict up to 96% of all the existent matches in a 5k dataset.

The Random Forest algorithm is at the theoretical maximum performance given the feature selection and tuning, showing very little room for future improvement with the same data structure as training data. The current state is still not acceptable in production as, even though there is some room for manual matching of false *non-matches*, there is no tolerance for any false *matches*. This could be mitigated by evaluating the prediction probabilities outputted by the model, but it falls off

the scope of this study. The tuning of the algorithm did not show a significant increase of the AUC of the precision recall curve but by trying to maximize the F0.5 score, the end result was closer to a production available model for the reconciliation as the true *non-matches* have a heavier weight in the bank's end goal.

Being that only a single reconciliation was evaluated, there is not definite proof the results would hold on the rest of the set, but the pipeline shows promising results. The Random Forest algorithm can be easily integrated in the current research team's workload due to its concept similarity to supervised classification methods used right now in pre-production. This and the fact that the process show greater sensitivity to the feature engineering changes, makes the process analogue to the current system where the team of business analysts' inputs produce the biggest results in terms of matching accuracy.

Chapter 7

Conclusions

The goal of transforming the current overfitted rule system into a supervised record linkage pipeline capable of similar performance was achieved. There are still some doubts regarding the viability of deploying it to production of a module of the reconciliation system. The percentage of false *non-matches* is low enough to entertain the idea of adding a layer of clerical review, but the existence of false *matches*, even if in low numbers, prevents this from being a turnkey solution to the problem.

The solution was designed to accommodate the current workflow and concern separation in the various steps of the process. Having these results while removing around 80% of the source relationship knowledge, shows promising future iterations of the process while leaving the fine tuning of the indexing and comparison steps in the hands of employees with the same set of skills used in the tuning of the current rules. This allows for the supervised classification to be used as a complement of the manual workload, redirecting the current effort and reducing the overfitting of each rule set. This is on par with Natixis needs as the automation factor works as a enhancer of the manual work, not a replacement.

7.1 Current R&D Blocking Points

Along the process of studying the problem of the reconciliation system and a possible suitable solution, a lot of barriers were encountered. The bank is actively deploying resources and investing in research, but the current platform of the Portugal office is not well suited for the effect. In order to take advantage of the current workforce innovation drive, changes should be made in terms of streamlining access to information. This can be achieved by easing the bureaucratic process for research projects or having a sandbox environment capable of performing data analysis and transformation tasks. Other than that, policy on software and data structures documentation should live up to the standards of commercial software for all banking support activities. Missing documentation and structure rules makes the task of studying a specific system very reliant on multiple interviews with the people actively working on it where a lot of information can be left on the table. This also results in the storage of noisy data where there is an expectation of consistency.

7.2 Future work

The next step into evaluating the reliability of the solution presented is to apply the same naive cleaning of the rule sets of multiple reconciliations, across at least one of each provider. The results can be a good indication on the direction to follow. If the record linkage approach proves to be robust to the differences between data extraction on multiple providers, the pipeline can be improved by small increments in relationship details.

The final goal will be to achieve a balance between the least worker input possible on the indexing and comparison steps, in the form of thresholds for comparison and adequate comparison algorithms, while trying to achieve a false *match* rate as close as possible to 0.

References

[1] Vera Cardoso Ferreira Aiken, João Ricardo Rebouças Dórea, Juliano Sabella Acedo, Fernando Gonçalves de Sousa, Fábio Guerra Dias, and Guilherme Jordão de Magalhães Rosa. Record linkage for farm-level data analytics: Comparison of deterministic, stochastic and machine learning methods. *Computers and Electronics in Agriculture*, 163:104857, 2019.

[2] Akiko Aizawa and Keizo Oyama. A fast linkage detection scheme for multi-source information integration. In *International Workshop on Challenges in Web Information Retrieval and Integration*, pages 30–39. IEEE, 2005.

[3] Daniel Berrar. Performance measures for binary classification. 2019.

[4] Mikhail Bilenko, S Basil, and Mehran Sahami. Adaptive product normalization: Using online learning for record linkage in comparison shopping. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[5] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. Adaptive blocking: Learning to scale up record linkage. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 87–96. IEEE, 2006.

[6] Mikhail Bilenko and Raymond J Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, 2003.

[7] Mikhail Bilenko and Raymond J Mooney. On evaluation and training-set construction for duplicate detection. In *Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pages 7–12, 2003.

[8] Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):1–50, 2016.

[9] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[12] Monica Scannapieco Carlo Batini. *Data Quality: Concepts, Methodologies and Techniques*. Data-Centric Systems and Applications. Springer, 2006.

[13] Supriya Chakraborty and Nabendu Chaki. A survey on the semi-structured data models. In *Computer Information Systems–Analysis and Technologies*, pages 257–266. Springer, 2011.

[14] Zhaoqi Chen, Dmitri V Kalashnikov, and Sharad Mehrotra. Exploiting context analysis for combining multiple entity resolution systems. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 207–218, 2009.

[15] P Christen. Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. 2012.

[16] Peter Christen. Automatic training example selection for scalable unsupervised record linkage. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 511–518. Springer, 2008.

[17] Peter Christen, Tim Churches, et al. Febrl-freely extensible biomedical record linkage. The Australian National University, 2002.

[18] William W Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems (TOIS)*, 18(3):288–321, 2000.

[19] William W Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480, 2002.

[20] J De Bruin. Python Record Linkage Toolkit: A toolkit for record linkage and duplicate detection in Python, December 2019.

[21] Mohamed G Elfeky, Vassilios S Verykios, and Ahmed K Elmagarmid. Tailor: A record linkage toolbox. In *Proceedings 18th International Conference on Data Engineering*, pages 17–28. IEEE, 2002.

[22] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[23] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[24] Shanti Gomatam, Randy Carter, Mario Ariet, and Glenn Mitchell. An empirical comparison of record linkage procedures. *Statistics in medicine*, 21(10):1485–1496, 2002.

[25] Lifang Gu, Rohan Baxter, Deanne Vickers, and Chris Rainsford. Record linkage: Current practice and future directions. *CSIRO Mathematical and Information Sciences Technical Report*, 3:83, 2003.

[26] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[27] Mauricio A Hernández and Salvatore J Stolfo. The merge/purge problem for large databases. *ACM Sigmod Record*, 24(2):127–138, 1995.

[28] Mauricio A Hernández and Salvatore J Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data mining and knowledge discovery*, 2(1):9–37, 1998.

[29] W Philip Kegelmeyer. Smote: Synthetic minority over-sampling. In *Knowledge Based Computer Systems: Proceedings of the International Conference: KBCS–2000*, page 46. Allied Publishers, 2000.

[30] Heikki Keskustalo, Ari Pirkola, Kari Visala, Erkka Leppänen, and Kalervo Järvelin. Non-adjacent digrams improve matching of cross-lingual spelling variants. In *International symposium on string processing and information retrieval*, pages 252–265. Springer, 2003.

[31] Hanna Köpcke and Erhard Rahm. Training selection for tuning entity matching. In *QDB/MUD*, pages 3–12, 2008.

[32] Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210, 2010.

[33] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, 2006.

[34] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.

[35] Vladimir Levenshtein. Binary codes capable of correcting spurious insertions and deletion of ones. *Problems of information Transmission*, 1(1):8–17, 1965.

[36] Matthew Michelson and Craig A Knoblock. Learning blocking schemes for record linkage. In *AAAI*, volume 6, pages 440–445, 2006.

[37] Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

[38] Sreerama K Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998.

[39] Felix Naumann and Melanie Herschel. An introduction to duplicate detection. *Synthesis Lectures on Data Management*, 2(1):1–87, 2010.

[40] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[41] Howard B Newcombe, James M Kennedy, SJ Axford, and Allison P James. Automatic linkage of vital records. *Science*, 130(3381):954–959, 1959.

[42] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[44] Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, 23(4):3–13, 2000.

[45] Jason D Rennie, Lawrence Shih, Jaime Teevan, and David R Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.

[46] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[47] Yanmin Sun, Andrew KC Wong, and Mohamed S Kamel. Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, 23(04):687–719, 2009.

[48] Sheila Tejada, Craig A Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.

[49] Sheila Tejada, Craig A Knoblock, and Steven Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 350–359, 2002.

[50] Brigitte Van Berkelt and Koenraad De Smedt. Triphone analysis: A combined method for the correction of orthographical and typographical errors. In *Second Conference on Applied Natural Language Processing*, pages 77–83, 1988.

[51] William E Winkler. Frequency-based matching in fellegi-sunter model of record linkage. *Bureau of the Census Statistical Research Division*, 14, 2000.

[52] Norman Zerbe, Christopher Hampf, and Peter Hufnagl. Extension of the identity management system mainzelliste to reduce runtimes for patient registration in large datasets. In *Artificial Intelligence and Machine Learning for Digital Pathology*, pages 228–245. Springer, 2020.

[53] Huimin Zhao and Sudha Ram. Entity identification for heterogeneous database integration—a multiple classifier system approach and empirical evaluation. *Information Systems*, 30(2):119–132, 2005.