N°d'ordre: .....

République Algerienne Démocratique et Populaire

Ministère de l'enseignement supérieur et de la Recherche scientifique

Université Djillali Liabès de Sidi Bel Abbès

Faculté des Sciences Exactes

Département d'informatique

Laboratoire EEDIS

Thèse de Doctorat

en 3ième Cycle

Filière : Informatique

Spécialité : Technologie de l'information

Par

Mr Benkhaled hamid naceur

Qualité des données dans le contexte Big Data

"Data Quality in the Big Data context"

Soutenue le 15/02/2021. devant le jury :

Pr.	Zakaria ELBERRICHI	UDL SBA	Président du jury
Dr.	Farah BEN NAOUM	UDL SBA	Examinatrice
Pr.	Mimoun MALKI	ESI	Examinateur
Dr.	Djamel AMAR BENSABER	ESI	Examinateur
Dr.	Berrabah Djamel	UDL SBA	Directeur de thèse

Année Universitaire : 2020. - 2021.

To my familly…

Acknowledgement

Above all, I thank God, the almighty, whom I bless for having finished this work in the best possible conditions.

I would like to express my sincere gratitude to my advisor Prof Djamel Berrabah for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

I would also like to thank Prof Faouzi Boufares for his consistent support and guidance during the running of this project.

I'd like to express my thanks to the examiners for accepting to be part of the jury members and for evaluating my work.

As a preamble to this brief, I wish to express my sincere thanks to the people who have brought me their help and contribute to the development of this thesis as well as to the success of this great University Year.

I will not forget my parents for their contribution, their support, and their patience.

Thanks, everyone.

Saida, February 25, 2021.

Contents

List of Figures

List of Tables

General Introduction

1.1 Context

Over the last decades, organizations around the world have become more and more dependent on the decision support systems (DSS). Most of the enterprises are competing on implementing the best ones to have better decisions and a better place in the market. Before the arrival of Big Data, Data warehouse (DW) systems proved their efficiency as the best DSS. They always helped the enterprises to make the right decision and take a step ahead over the competitors. However, recently, we are witnessing the Big Data era. Big Data analytics is becoming more and more a reliable decision support system, specifically, with the huge adoption of social media and smartphones, which generates a huge amount of data that the traditional DWs cannot deal with.

Besides, although the DW systems and Big Data analytics are considered now as the best decision support systems, these two technologies can sometimes fail to meet the stakeholder's expectations. In fact, many wrong decisions have been made causing bad consequences reaching even project delays and interruptions due to poor Data Quality.

Annually, in the US alone, 3.1 trillion dollars is the cost of bad data quality according to IBM (International Business Machines) [Redman 2016] where most of the data stored in the organizations' databases is erroneous [Geiger 2004]. As a result, the stakeholders can lose their trust in the decision support systems and look for other solutions. Consequently, the enterprises were more aware of the importance of Data Quality. Most of them invest a huge amount of money to resolve the anomalies of their stored data and make it usable to get a piece of information out of it. Data quality problems can appear in different ways. Indeed, there are problems of completeness (missing values), duplicate values, referential integrity problems ,and many other anomalies.

In this thesis, we focus on the problem of duplicate values and specifically on the Record Linkage (RL) problem. Record Linkage is

the process that aims to detect all the records that refer to the same real-world entity and then merge them into a single tuple. RL can also be found under the names (Entity resolution [Ouhab *and al.* 2017], Deduplication [Sarawagi & Bhamidipaty 2002], Merge-Purge problem [Hernández & Stolfo 1995], and the matching problem). Recently, the Record Linkage process was exploited in several domains such as linking data for health services, also to detect duplicates in the bibliographic citations, analyzing census data, privacy-preserving, and fraud detection.

The perfect way to apply record linkage over dirty data is to compare each record in the dataset to all the others which is the Cartesian product of the data. Unfortunately, doing that could end-up with billions of comparisons, which is not reasonable in the case of a large dataset.

In order to reduce the number of comparisons in the RL process, the blocking technique is used. Usually, blocking consists of creating a set of blocks. Each block will group the records that share a common value named "Blocking Key Value (BKV)" (attribute field's value). In that way, only the records grouped in the same block are to be compared to each other. The record linkage community proposed several blocking techniques. These techniques are discussed further in this thesis.

1.2 Problematic

Dealing with duplicate values using the RL process is a very challenging field Several problems have to be taken into consideration, specifically, when the data is gathered from different sources, each one with its own schema and data representation. The major challenge is to deal with data without identifiers (Primary Keys) which makes it necessary to compare the other attributes in order to check the similarity of the records.

When data is collected from different sources, the same attribute might be represented differently in two different sources. For example, in one data source we can find the sex attribute as a binary value (0/1) while in

another one as (M/F). As a result, data standardizing is a very important step for a successful RL process. In general, the challenges that have to be resolved for an efficient RL process are the following:

1.2.1 Indexing/Blocking

In order to reduce the search space, blocking techniques are used. Several blocking approaches were proposed in the literature, each one with its advantage and drawbacks. Choosing the right blocking technique for a specified problem is very important depending on the type of data and the quality of it.

With the arrival of Big Data, the traditional blocking techniques are becoming obsolete A lot of them cannot handle the huge amount of data nowadays and take a lot of executing time. As a result, the record linkage community tried to adapt the traditional blocking techniques to the new Big Data requirements by using distributed computing architectures like cloud computing and the famous programming model Map-Reduce. So, proposing an efficient blocking technique and a scalable one that can deal with the new Big Data requirements is very challenging.

1.2.2 Blocking keys selection

Two important parameters control the performance of a good blocking technique. The first one is the blocking key value. A blocking key can be formed using one field (attribute) or a concatenation of several parts from a set of fields. For example, a BKV can be formed using the First-Name value or by the concatenation of the first three characters from the First-Name field and the ZIP code from the address field. Table 1.1 shows an example of blocking keys generating from the restaurant dataset. Two blocking keys were generated. The first one (BK1) is the Soundex phonetic encoding of the restaurant's name concatenated with the phone number. The second one (BK2) is the NYSIIS phonetic encoding of the restaurant's name concatenated with the address number.

Table 1.1 – *A Blocking keys example from the restaurant dataset*

BK1	BK2	Name	Address	City	Phone	Type
A6553102461501	LASANG435	arnie morton's of Chicago	435 s. la cienega blv.	Los Angeles	310/246-1501	American
H3413104721211	STADAC12224	art's deli	12224 ventura bold	Studio city	818-762-1221	delis

The second important parameter is the number of blocking keys, using more than one single blocking key can improve the efficiency of the blocking techniques since the blocking keys values will contain more information about the record. Consequently, choosing the best fields to form a good BKV and the number of blocking keys is a very important step to get the best result out of the RL process.

Selecting the fields to form the blocking key values is a task that usually has to be done manually by a domain expert. Proposing an automatic blocking key selection that do not need the intervention of a domain expert is very challenging.

1.2.3 Block sizes control

Recently, several applications in different domains require fast queries' responses in order to make real-time analyses. This leads us to the real-time Record Linkage where finding duplicates for a given record in a query has to be done is a specified time [Fisher *and al.* 2015]. Even when using one of the existed blocking techniques in the literature, real-time RL can sometimes fail to return the query's result in the desired time. In fact, the existing blocking techniques can sometimes generate very large blocks which leads to high computational time. As a result, controlling the maximum size of the generated blocks is a very important condition for a successful real-time RL. On the other hand, generating very small blocks can reduce the effectiveness of the RL process, possible matches can be assigned to different clusters which leads to undetected true matches. Privacy-preserving RL is also a domain that requires a specified minimum size of blocks. As example, the k-anonymity approach is based on the fact that each block needs to contain at least K records [Karakasidis & Verykios 2012]. As a result, proposing a mechanism to control the size of the generated blocks is also very challenging.

1.2.4 Non-Latin languages

Most of the proposed RL approaches do not take into account the Arabic language, a lot of them are proposed to be used with Latin characters which is the case of the English language. Proposing a solution to adapt the existing RL approaches for non-Latin characters as the Arabic language is one of the fields that didn't get enough attention in the RL community. Proposing an Arabic adapted RL approach has become a pervasive challenge for the Arabic language community.

1.3 Contribution

Based on the RL challenges discussed in the previous section, our contribution in this thesis covers all the aforementioned points. A solution is proposed for each of the previous challenges. Our propositions can be summarized into the following points:

1. As a first step, we studied data quality in the most important decision support systems that are Big Data and Data warehouses [Benkhaled *and al.* 2020b] [Benkhaled & Berrabah 2019]. This study allowed us to have an overview about the existing approaches for managing data quality in the literature and in which domain a contribution should be proposed.
2. We propose a novel approach in the field of Record Linkage [Benkhaled *and al.* 2019]. The proposed approach is based on the K-Modes algorithm as a blocking step. K-Modes divides the data into a set of blocks using only the blocking keys as clustering attributes instead of using all the dataset attributes. Doing that helps to improve the execution time and avoid using attributes of bad quality in clustering. Our choice of using the K-Modes algorithm is its ability to cluster categorical data which was not the case of other clustering algorithms like K-Means. Using K-Modes helped as also avoiding the data numerical conversion step which can be very costly in the case of large datasets. Once the blocking is done, a filtering technique is executed in order to remove more unlikable

pairs from the comparison step. For the matching step, a set of string similarity metrics were evaluated (Edit distance, Jaro-Winkler, and Jaccard).

3. In order to adapt our solution to the new Big Data challenges. We propose a parallel implementation of our K-Modes based RL using the Map-Reduce paradigm [Benkhaled *and al.* 2019] . Two main Map-Reduce jobs are introduced, the first job named IBPK (Indexing Based on Parallel K-Modes). IPBK is based on a parallel implementation of the K-Modes algorithm using Map-Reduce. IBPK receives the blocks of the dataset as input and divides them to a set of clusters where all the records that are close to each other in terms of similarity are brought together. When the clustering is done, PMF (Parallel matching with filtering) does the matching step in a Parallel architecture, all the records with the same cluster ID are directed to the same reducer where the filtering and the matching steps are executed.

4. We propose a novel unsupervised approach for the automatic blocking keys selection [Benkhlaed *and al.* 2020]. In this approach, we treat this problem as a feature selection one. The solution is based on the recently proposed meta-heuristic Bald Eagle Search (BES) optimization algorithm [Alsattar *and al.* 2019]. Meta-heuristics have been widely used for the problem of feature selection as it is considered as an NP-Hard problem. To the best of our knowledge, no work in the literature used meta-heuristic algorithms to solve the problem of automatic selection of blocking keys. The proposed solution helped automatizing the selection of blocking keys used in the blocking step.

5. We propose a solution to control the size of the blocks generated by the K-Modes algorithm [Benkhaled *and al.* 2020a]. This solution helps to improve the efficiency of the proposed K-Modes based RL

in the case of real-time applications and privacy-preserving RL. The main points of this approach are: (1) generates the blocks using the K-Modes based record Linkage. (2) Merge the generated blocks that are under a predefined minimum size and the similarity between their modes is maximized. (3) Divide the blocks that are larger than a predefined maximum size using another blocking key with the standard blocking.

6. In order to apply our proposed K-Modes based RL on non-Latin languages like the Arabic language, a solution is proposed by adding an additional transliteration step during the first step of RL which is data standardizing.

1.4 Organisation

The rest of this thesis is organized as follows: (1) Chapter 2 provides a background about the most important concepts and techniques in this thesis. It defines Data warehouses, Big Data, data quality generally and Record Linkage particularly. Chapter 3 presents the current Record Linkage state of the art and provides a comparative study based on the main RL challenges. Chapter 4 presents the proposed approaches with details. The experiments and results are presented in Chapter 5. Chapter 6 concludes the thesis with some of the perspective works.

Background

Sommaire

2.1 Introduction

In this chapter, a background is provided about the preliminaries and technologies used in the rest of this thesis. A general overview of the best two decision support systems that are used by most of the worldwide enterprises (DW systems and Big Data analytics) is provided in sections 2.2 and 2.3. In section 2.4 we define the data integration process. Section 2.5 discusses the impact of data quality on decision support systems. Finally, the record linkage process is presented in section 2.6.

2.2 Data warehouse systems

Before the Big Data era, data warehouse systems were considered as the most effective tool for decisions support, William H. Inmon defines a data warehouse as "a collection of Integrated, Subject-Oriented, Non-Volatile and Time-Variant data in support of management's decisions" [Inmon 1992]. In this definition, four important points were mentioned. Firstly, Integrated means that the DW can be viewed as an integration system that integrates data collected from heterogeneous sources under different formats into a unique representation. Subject-Oriented stand for the fact that the DW systems are used in a specific area analysis. Non-volatile means that once the data get stored in the warehouse it can't be changed. Finally, time-variant since time is always considered as a dimension in the DW systems. Data warehouse's effective analysis results make organizations around the world more and more reliant on it in making strategic decisions [Halevy *and al.* 2005].

2.2.1 Data warehouse architecture

The architecture of a data warehouse is composed of four principal layers. This architecture is depicted in Figure 2.1 , in which we can see the organization of the four layers.

— **Data sources:** The data warehouse stores data from multiple sources. This stored data is of different formats. For example, (1)

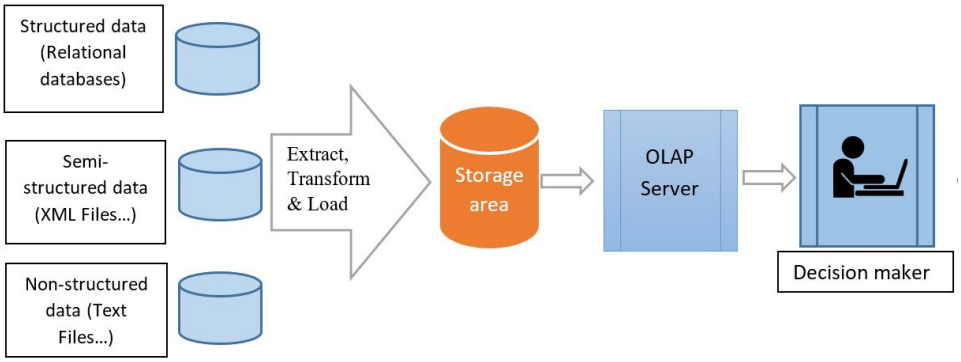

Figure 2.1 – *Data warehouse general architecture*

Structured data like the relational database of an enterprise. (2) Semi-structured data like the one gathered from the Web under an XML formats and (3) non-structured data which are the text files like emails and tweets.

— **Storage area:** It is where the data get stored after the extraction from multiple sources. Generally, these data have to be cleaned from errors first which is known as the ETL process (Extract, Transform, and Load) (see next sections).

— **OLAP server:** The OLAP (Online Analytical Processing) server allows the user to access the data warehouse and to express analytical queries.

— **Front end tools:** According to the final user's needs, the data get formatted so it can be used for a specific problematic.

2.2.2 Multidimensional Modeling

As mentioned in the previous section, the principal purpose of using a data warehouse is to make a decision or the extraction of hidden information. Generally, decision applications based on data warehouses systems use OLAP as an analytical process. In order to facilitate this process, the data stored in the data warehouse is modeled in a multidimensional way. R. Kimball popularizes the multidimensional modeling in the '90s. Since

that, multidimensional modeling becomes the most suitable type of modeling for decision support and analysis. In this type of modeling, the data get structured and organized to support analysis and offer a multidimensional view. Multidimensional modeling contains two important concepts: Facts and dimensions. A fact is the representation of the analyzed subject. It is composed of measures that represent all the needed information for decision-making. Each dimension contains a set of attributes. These attributes can be atomic or non-atomic. In the case where the attributes are atomic, the dimensions can be transformed to a hierarchy of relations. For example, the dimension "date" can be structured as a year, quarter, month, week, and day [Khouri 2013] [Chaudhuri & Dayal 1997].

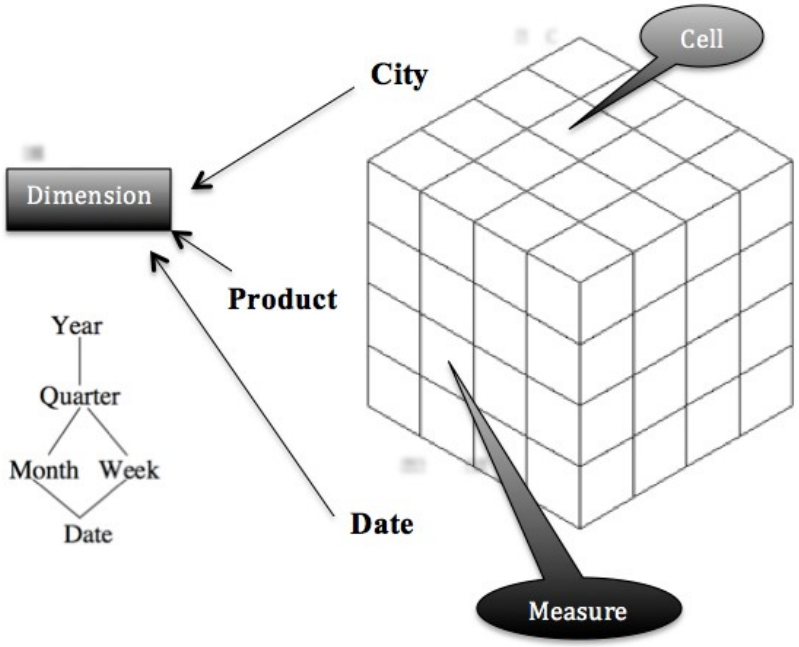

Figure 2.2 – *Multidimensional representation of data*

Figure 2.2 shows an example of a multidimensional modeling using a data cube. In this example, each cell of the cube represents a measure and each axe of the cube represents a dimension. An example of a time hierarchy is also depicted in the figure.

In order to ease the analytical process, OLAP offers a set of predefined operations for the user to use them in his analytical queries and data analyzing. In what follows, a brief definition of the most used OLAP

operations [Kämpgen *and al.* 2012].

— **Roll-Up:** This operation allows climbing up in the hierarchy of a dimension. For example, the dimension date will viewed by years instead of months.

— **Drill-Down:** It represents the inverse of Roll-Up, it is used to show more detailed data by stepping down a hierarchy of a dimension.

— **Slicing:** it is the extraction of a sub cube from the principal cube using a selection of one dimension.

— **Pivot:** also known as the rotation operation, it is used to view from a different perspective, the rotation is done around an axis.

Other OLAP operation exists, like: Dicing, Scoping, Screening, Drill across, Drill trough, split, push, and switch. Figure 2.3 provides a visual demonstration of some operations.

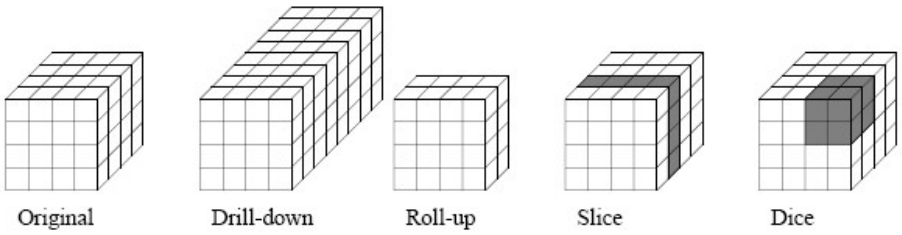

Figure 2.3 – *OLAP operations*

2.2.3 ETL (Extract, Transform, and Load)

The collection of data from multiple sources in different formats, the cleaning and the transformation of the collected data to be loaded correctly in the data warehouse is known as the ETL process (Extraction, Transformation, Loading) [Jensen *and al.* 2010] [Trujillo & Luján-Mora 2003]. ETL is considered as the most important process in the data warehouse life cycle. It represents 70% of the efforts in the data warehouse

projects [Liu *and al.* 2012]. The process deals generally with a huge amount of data which makes it an extremely time consuming process [Berkani *and al.* 2013]. ETL is implemented as a workflow where data processors are connected by data flows [Patil *and al.* 2011]. Generally, the most important tasks in an ETL workflow are [Trujillo & Luján-Mora 2003]:

1. The selection of the different sources for the extraction. Generally, the collected data is heterogeneous (Text files, Relational databases, XML files, Web data ... etc.) and stored in different systems.
2. Once the data is extracted from the selected sources, transformations functions are applied to the extracted data. Usually, the most relevant problems to deal with at this stage are: Duplicate values, missing values, surrogate key assignment, and checking referential integrity.
3. Mapping of the extracted attributes to the target attributes (Data warehouse attributes).
4. Loading the data into the warehouse.

The most important phase in the ETL process is the transformation phase, also called the data staging area (DSA) [Vassiliadis *and al.* 2002]. Most of the data cleaning tasks are performed at this stage in order to improve DQ. However, ETL tools do not include advanced cleaning capabilities [Trujillo & Luján-Mora 2003]. As a result, data of poor quality can appear causing serious issues in decision making by giving wrong conclusions.

2.2.4 Conceptual modeling for the ETL Workflow

After knowing the importance of the ETL process during the data warehouse life cycle, proposing a conceptual model for the ETL process can help decreasing the cost of the development and maintenance of the data warehouse. This next section is dedicated to the existence approaches in the literature for modeling an ETL workflow.

The conceptual model has a great role in facilitating the maintenance and the transformations phase of an ETL process. It helps also in identifying all the business entities that have a part in the process, defining a formal representation of these entities and describing the business requirements. In the last few years, designing a conceptual model for an ETL workflow was a hard task due to the lack of standards and guidelines and the non-existence of a specific model. A lot of effort was invested inside the ETL community by proposing a number of methods including graphs, UML, ontologies and BPMN.

UML-Based approach

In the last years, UML has stood as a powerful tool in the world of software engineering. It gives the possibility to create models that help in visualizing a system in a clear and standardized way. J Trujillo and S Lujan-Mora proposed an approach for modeling an ETL process using Unified Modeling Language (UML) [Trujillo & Luján-Mora 2003]. The authors proposed a standard representation for the most common and used operations in the ETL workflow, like the integration of multiple data sources, the generation of a surrogate key and converting data types and more. This approach is based on the decomposition of complex ETL processes into simple ones. It's also mentioned that this approach fits perfectly with structured data, and to apply this approach over simple files (txt files. . . Etc.). A Wrapper has to be used to transform any source into an attribute-based source. The ETL mechanisms have been reduced to reduce the complexity of this approach but they maintain the most used and critical mechanisms.

Graph-Based approach

One of the first approaches that were proposed in the ETL community for the conceptual modeling of the ETL workflows was based on graphs. P Vassiladis and al, proposed a conceptual approach using graphs focusing

on the interrelationships between attributes, concepts and the needed transformations during the loading part [Vassiliadis *and al.* 2002]. The authors proposed a set of different transformations needed for different ETL scenarios. They didn't use UML standards for representing concepts and attributes since attributes have to be treated as first-class citizens to show the interrelationships and the mapping between them. A three-layer architecture was proposed in their approach for the conceptual modeling of ETL workflow:

Schema layer: this layer contains a set of specific ETL scenarios, all the entities in this row are instances of the Meta-model layer.

Meta-model layer: contains the classes of all the aforementioned instances; using these classes an ETL developer can represent any ETL scenario.

Template layer: this layer contains specific constructs (subsets of the meta-model layer). These constructs are also Meta-classes, but they are customized for more specific ETL scenarios.

BPMN-Based approach

Business process modeling notation (BPMN) is a Standard used in modeling business processes. It gives to the users the possibility to model any business process separately from the used tools. BPMN is understandable for all the parts in the business entities (technical developers, analysts . . . and more) which is the major advantage of using BPLN in modeling. [El Akkaoui & Zimányi 2009] proposed a conceptual approach for modeling an ETL workflow independently from the used platform and based on BPMN, they demonstrated how to implement the obtained model using BPEL (Business Process Execution Language).

The authors mentioned that any ETL process could be considered as a specific type of business process and showed how BPMN can be adapted in order to design an ETL workflow. They proposed an ETL palette com-

posed of a set constructs and they regrouped these constructs into four categories. For example, an ETL task is represented in their approach by the BNMP activity or a sub-process and they used BNMP getaways and events to represent ETL control object that manages the control sequences. BNMP artifacts in this approach are dedicated to represent data annotations and getaway conditions. At the final point, the authors showed that an ETL process expressed in BPNM can be easily transformed into an executable specification using BPEL.

In addition to the aforementioned approaches, other approaches were proposed in the literature like the ontology-based approach [Skoutas & Simitsis 2007].

2.3 Big Data

Before starting understanding Big Data and the technologies that have to be mastered in order to extract a piece of useful information from this huge amount of data, let's take a look on some statistics about it:

Until 2003, 5 Exabytes was the amount of the generated data in the world. Now, this amount of data is generated in just TWO days [Sagiroglu & Sinanc 2013]. The world expected that by the end of 2012 the amount of data generated will be around 2.72 Zeta bytes (ZB) and it will be doubled every two years, reaching 8 ZB in 2015 [Guide 2013]. However, in 2015 the amount of the generated data reached 15 ZB , this shows the unacceptable rate the that data is growing with.

Cisco is considered as one of the biggest network companies, they analyze the data traffic of each year. The company mentioned in their white paper that in 2018 we were dealing with 17 EB of data and they predicted that by the end of 2021 we will reach 47 EB of data and that's a lot of zeros to be handled by traditional databases systems [Cisco 2016].

Cisco also mentioned the major reasons behind this huge explosion in the amount of data. The first one is smart phones. Now, everyone has a

smart phone with its applications, each of this applications is generating a lot of data in a daily manner. Another reason mentioned by Cisco is cell networks (3G, 4G, 5G) using this networks a user can generate data anywhere and at any time.

Big data has a lot of growth drivers, the principal ones are social media. For example, every 60 seconds we have 4 million likes on Facebook, 300 000 tweets on Twitter, 300 hours of videos are uploaded on YouTube with 4 billion views every minute, and 1 million users' likes on Instagram.

2.3.1 Big Data definitions

In the last years, researchers and enterprises have defined the term "Big Data" in a different and contradictory ways. Some focused on the volume of the data and the others focus on the technological techniques that are used in order to analyze and extract important information.

The first definitions of Big Data were proposed by the GARTNER group in 2001. The group proposed a three Vs definition model: Volume, Velocity, and Variety. The first V for the growing amount of data, the second one for the speed rate that the data in increasing with and the third one for the different type of the collected data. This definition was updated later in 2012 by the GARTNER group and extended by IBM by adding a fourth V that stands for Veracity.

Oracle in 2013 defines Big Data without using any Vs [Dijcks 2012]. They define it as traditional Enterprise data in addition to sensor data, Weblogs, and data from social media like Facebook Twitter and more. Same as the previous definition, this one gives no numerical specification on when we should use the term Big Data.

Intel defines Big Data using numerical statistics from their experiments with their partners [Guide 2013]. In their definition, Intel associate the term Big Data with the enterprises that generate a huge amount of date weekly (300 Terabytes TB).

Microsoft defines Big Data in a curt way "Big data is the term increasingly used to describe the process of applying serious computing power, the latest in machine learning and artificial intelligence, to seriously mas-

sive and often highly complex sets of information" [Redmond 2012]. In this definition, Microsoft introduced a new point of view by mentioning machine learning and artificial intelligence technique that are applied in order to extract information from huge Big Datasets. This definition introduces the fact that technologies can form a crucial part of the Big Data definition.

2.3.2 The V model

In the previous section, we discussed the existed definitions for the term Big Data in the literature. The most used and accepted definition by the Big Data community is the V model definition (Figure 2.4). In this section we will provide a brief description for each concept, we will start with the first three Vs (Volume, Variety, Velocity) that were proposed by the Gartner Group in 2001 then we will define the fourth one that was added by IBM (Veracity) [Bello-Orgaz *and al.* 2016] [Zikopoulos *and al.* 2013].

Volume: The term volume is used to refer to the huge amount of data collected from different sources (mobiles, social media, sensors . . . Etc.). The benefit of gathering these big large data sets and analyzing them is to extract the most possible knowledge for researchers and companies [Bello-Orgaz *and al.* 2016]. In the recent years, data moved from been measured using petabyte to zeta-bytes (2009: 0.8 ZB, 2011: 1.8 ZB, 2018: 35 ZB) [Zikopoulos *and al.* 2013].

Variety: Big Data analytics use different types of data. This data is collected from different sources like social media, mobile phones, and sensors. It can be structured like traditional relational databases or semi-structured or unstructured data like text files. So, the term variety is to describe the different types of collected data.

Velocity: IBM describes the velocity as one of the most miss understood characteristics of Big Data. They define it as the speed in which the data arrives to an enterprises and how much time it took to be analyzed

and well understood.

Veracity: Sathi and Arvind define veracity using two important points [Sathi 2012]. Data credibility and suitability for the target audience. Big data is collected directly from business companies without being supervised by specialists. Consequently, it might be full of errors. So, data credibility is an important point in order to get the right conclusions. it's also mentioned in their definition of veracity that data privacy and legal concerns have to be taken into consideration.

In addition to the aforementioned V definitions, some other researchers extend this V model by adding other Vs that define Big Data from other aspects [Gani *and al.* 2016]. For instance, Variability is defined as the problem of dealing with the peaks in the data flow that can cause some serious inconsistencies in the data loading phase, specifically, when dealing with data collected from social media and smartphones. Moreover, the Value of the collected data is very important in a Big Data analytic system. In fact, it is very important to know how much a large amount of data influence the value of insights and benefits of an Enterprise. The complexity of dealing with this huge amount of data is also very challenging, the collected data has to be cleansed, linked and connected, if the complexity is not considered then the BD analytic system is not well organized.

2.3.3 Big Data analytics

Map-Reduce

In order to simplify processing large amounts of data in a distributed way, new programming models, and architectures with implemented algorithms were proposed. Map-Reduce is a programming model with an associated implementation [Dean & Ghemawat 2008]. The user has to specify only the map with the reduce function and the associated system will automatically manage: (1) the parallel calculation across several clusters. (2) Dealing with machine breakdowns. (3) The communication between the different nodes (4) Data aggregation. In the last years, Map-Reduce has stood as the most efficient Big Data solution [Bello-Orgaz *and al.* 2016].

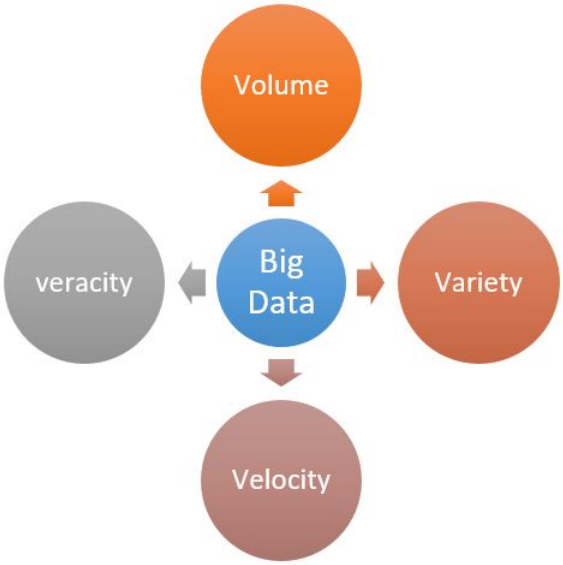

Figure 2.4 – *Big data Vs model*

Map-Reduce has two principal components: Mappers and Reducers. The role of the mappers is to generate an intermediate Key/value pairs as an output. The reducers take the mapper's outputs as input and merge the values of all the pairs with the same associated key. The idea behind using keys in the Map-Reduce approach is to make every mapper and reducers independent from the others. Doing that, the operation can be executed in parallel.

Generally, the three principal functions in the Map-Reduce paradigm are:

1. **Map ():** Before calling this function, the map-reduce system splits the data into equal parts and presents it as an input to the Map function at the level of each mapper. Each mapper applies the map function to its data and generates pairs of key/values.
2. **Shuffle ():** The data generated by the mappers is directed as an input to the reducers. The goal of the shuffling function is to regroup all the generated pairs with the same key at the same worker node.
3. **Reduce ():** The reduce function is executed over the data, which generates the final result.

The best way to understand the Map-Reduce paradigm is with the word count example (Figure 2.5). As shown in the example, once the data gets split over the Mappers, the map function is applied over it. In this case, the map function is associating with each word as a Value "1". The output of the mappers is a Key/Value pairs where the key is the word and the value is 1. When the mapping is done, the data get shuffled, all the pairs with the same key are directed to the same node where the reduce function is executed. In our example, the reduce functions is the summation of all the values with the same key.

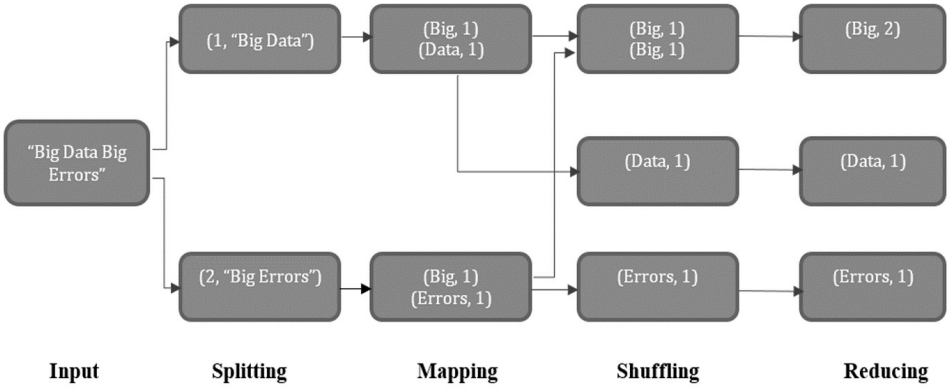

Figure 2.5 – *Words counting with Map-Reduce*

2.3.4 Hadoop

Hadoop is a framework that supports the Map-Reduce model, it allows storing and processing a huge amount of data in a distributed and parallel way. The Hadoop framework gives the developer the possibility to focus on the application side leaving the system handles the other details. Apache Hadoop is an open-source Java software based on the Map-Reduce architecture.

As mentioned before, Big Data presents two principal challenges. The first one is a storage problem. In order to solve that, the Hadoop framework provides us with HDFS (Hadoop Distributed File System). All the dumped data gets distributed over different machines that are interconnected via a network. In the Hadoop perspective, we call these machines clusters. Hadoop architecture is based on the Master/Slave model (see

figure 2.6). It's composed of three principal components (NameNode, DataNode, and SecondaryNode). The second challenge is processing. Hadoop use the Map-Reduce model in order to process data in a parallel way.

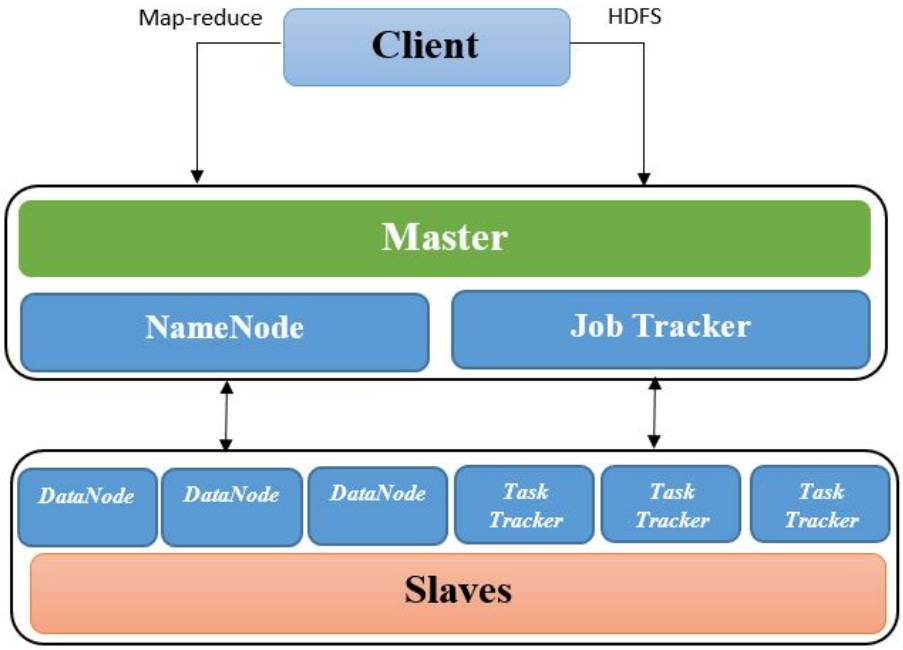

Figure 2.6 – *Master/Slave architecture*

NameNode: Know as the Master node. His job is to manage and maintain all the DataNodes (Slaves) by recording metadata like the information about data blocks, size of files, permissions. NameNode receives back reports from the DataNodes. These reports are known as Hear Beats.

DataNodes: These nodes represent the slave nodes. They are where data get actually stored. Their principal task is to serve reads and writes from the clients.

Secondary NameNode: The metadata inside the NameNode is maintained using 2 files (EditLog, FsImage). These files store all the modifications that have been made in the clusters since the NameNode creation. The first file (EditLog) contains recent information and the second one (FsImage) save all the information. The Secondary NameNode takes over

the responsibility of combining these two main files and to make the NameNode more available in case of a failure.

The main limitation of Hadoop is the fact that the reduce tasks can't start until all the mapping functions finish besides Hadoop is very costly in power consummation and CPU time.

Other implementations of the Map-Reduce programming model exists, such as Apache spark which is equipped with an extension of data sharing abstraction called "Resilient Distributed Datasets" [Zaharia *and al.* 2016]. Using this extension, Spark can easily include other processing workloads such as SQL, streaming, and machine learning.

2.4 Data integration

Data integration is one of the main challenges that an ETL process has to deal with during the DW life cycle. It consists of combining data coming from various sources in order to create a unified view of datasets which allows an easy analytical process [Doan *and al.* 2012]. Data integration also allows unified access to a company's data, the user does not have to access data of each department separately but the integration system will provide a unified and a cleaned view of data.

A real-world example of an integration case is depicted in Figure 2.7. In this example, we have records that represent publications in both bibliographic system ACM and DBLP. The integration system must detect that the records (conf/sigmod/RinfretOO01)/(375669) represent the same real-world entity (publication) and merge them during the data integration process. So the integration result will consist of 4 records instead of 5.

Another challenge that has to be dealt with during data integration is the different schema representation in each data source. The same data can be represented differently. For example, we can find the sex attribute in one dataset as a binary value (0/1) and in another one as (M/F). Schema anomalies are generally resolved using an intermediate table that maps

DBLP

Id	Title	Authors	Venue	Venue
conf/sigmod/RinfretOO01	Bit-Sliced Index Arithmetic	Elizabeth J. O'Neil, Denis Rinfret, Patrick E. O'Neil	SIGMOD Conference	2001
journals/sigmod/Hanson99	Statement from the Treasurer	Eric N. Hanson	SIGMOD Record	1999

ACM

Id	Title	Authors	Venue	Venue
375669	Bit-sliced index arithmetic	Denis Rinfret, Patrick O'Neil, Elizabeth O'Neil	International Conference on Management of Data	2001
306103	Building database-driven electronic catalogs	Sherif Danish	ACM SIGMOD Record	1998
304233	Managing Web data	Dan Suciu	International Conference on Management of Data	1999

Figure 2.7 – *Data integration example in the DBLP/ACM datasets*

each attribute of each source to its equivalent in the target table (final results). The intermediate table has to take into account the types of each attribute in all the sources.

Another important challenge is data fusion. When the system detects two or more records that refer to the same real-world entity, it's important to choose which one to keep in the final results or how to fuse all the records into a single one. Two approaches exist, instances based approaches, and meta-data based approaches [Bleiholder & Naumann 2009]. In the instances based approaches, the solution is to choose one of the variant attributes from the detected true matches records (choosing the maximum, the minimum or the average). In the meta-data based approach, the choice is based on the metadata of attributes. For example, the user can choose the most recent data as a priority.

2.5 Data quality

Although BD and DW systems have proven their standing over the years, they can sometimes fail to meet the stakeholder's expectations or give the right decisions. Indeed, many DW projects have been cancelled due to Data Quality (DQ) problems. DQ problems

can appears in different ways like missing values, duplicates records [Benkhaled *and al.* 2019][Ouhab *and al.* 2017] or the referential integrity problems. Poor quality data causes losses estimated at about 3 trillion dollar annually in the USA alone [Redman 2016]. Moreover, the Data Warehousing Institute in the US also mentioned that 15% to 20% of the stored data in most of the enterprises is of poor data quality [Geiger 2004]. Consequently, companies' leaders can lose their trust in the DW systems and look for other solutions since DQ problems can increase the cost of the Data Warehouse projects.

Data quality impacts on a typical enterprise can be summarized into three principal classes (Operational level, tactical level, and Strategic level) [Redman 1998].

Impact on operations: Poor data quality has a significant impact on the operational level. It can cause customers' dissatisfaction and increases the cost of a project. The principal objective of an enterprise is to make sure that all their clients are satisfied with the proposed services, but data errors can prevent that. For example, an error in a customer address will lead to a problem in product delivery and that means an unsatisfied customer. On the other side, poor data quality increases operational costs, since dealing with data errors, detecting, and correcting them costs more time and more budget (millions of dollars).

Impact at the tactical level: Poor data quality is more dangerous at the tactical level. Making a decision based on erroneous data can lead an enterprise to serious problems. Implementing a data warehouses is also more difficult with data of poor quality. The stored data have to be cleaned and corrected first which leads to additional fees, this can affect the budget of the DW project significantly. Another effect of poor data on the tactical level is the difficulty of re-engineering and putting the right data at the right place and time to serve the customers. Poor data can also increase the mistrust between organizations.

Strategic Impacts: Each enterprise has its own strategies and planning. When a strategy is deployed, specific plans have to take place and modified depending on the obtained results. If the results are wrong, late, or based on poor data, then the execution of the planned strategy is a daunting task.

2.5.1 Data quality management

Geiger defines data quality management as the process that defines policies and assigns roles so that data is collected, stored, and diffused [Geiger 2004]. He also mentioned that the business and the technology groups have to co-operate in order to achieve data quality management.

In the literature, there are several research studies aimed at identifying different anomalies in the data and their description schemes (definitions). These anomalies can be classified into two groups: metadata and data anomalies.

The manipulation of heterogeneous data was confronted with the fact that the descriptions were very poor if not missing. Metadata such as constraints and comments were almost non-existent. As a result, it was difficult to find the semantics of integrated data [Zaidi *and al.* 2015] [Zaidi *and al.* 2016] [Salem *and al.* 2014]. Data anomalies can be classified into three main categories, namely (1) intra-column anomalies (such as null values, outliers, syntactic and semantic abnormalities - failure to respect regular expressions -), (2) inter-column anomalies (semantic dependencies including functional dependencies, exact or approximate or conditional functional dependencies.) and finally (3) inter-line anomalies (duplicates and similar).

We will give, here, a synthetic overview. We studied the functionalities of data quality management tools and ETL tools such as Talend, Pentaho, Nadeef and Katara.

The comparison is based on several criteria. These represent features related to data quality such as (1) Statistical analysis of data: functions that provide simple statistics, for example, the number of lines, the number of null values, and the number of distinct and unique values. Statistics on data types (text, numeric and date) are used to analyze column characteristics, such as minimum, maximum, and average lengths; (2) The necessary transformations on the data values during integration: groups the functions for transforming dates and numbers; and (3) Duplicates and similar: different algorithms for calculating distances of similarities are implemented.

These tools offer the possibility to make statistics and data transformations. They allow the elimination of duplicates. However, it is the users who must guide them through this cleaning process. The user must know data structures and semantics to correct anomalies and inconsistent values.

These corrective actions must be initiated by the user. No help is provided to him. For example, Pentaho Data Integration and Data Cleaner do not allow functional dependencies to be verified, while Talend Data Quality does, but it is the user who must have knowledge of the data schema and the dependencies to be verified. It should be noted that no tools correct errors caused by the violation of functional dependencies. We have thus identified the weaknesses to be improved and the functionalities to be developed to contribute to the development of new tools that do not require the user to know the structures and semantics of the data manipulated from the sources and allow us to assist the correction of all types of anomalies. The rediscovery of metadata then becomes our objective to better address data quality issues. The aim is to discover the meaning and constraints that could be defined in each column, the relationships between the columns and to deduce the key columns in order to better achieve the deduplication.

Data quality dimensions:

In order to give a measured value for data quality, a set of data quality dimensions have to be used. Batini and Scannapieco divide the major data quality dimensions into two groups: principal and secondary. The principal DQ dimensions include Accuracy, Completeness, Currency, and Consistency. The secondary dimensions include accessibility, interpretability and other time-related dimensions. Moreover, one or two metrics are defined for each dimension [Batini & Scannapieco 2016].

The accuracy dimension can be defined syntactically and semantically. Most of the data quality methodologies take into account only the syntactic accuracy. It is defined as how close a value V to an element of the domain D. Several metrics exists in the literature to measure the syntactic accuracy such as the Edit distance, similar sounds (Like Soundex and NYSIIS) and character transposition. Completeness is generally represented by the presence of the null value in the data collection where the value exists in the real-world. We can find different types of completeness as: value, attribute, and tuple and relation completeness.

Currency, Timeliness, and Volatility are the most important time-related data quality dimensions. Currency refers to the rate in which the stored data gets updated with. Usually, currency is measured using the last update metadata. Volatility is related to the type of data, it is low if the data is stable like the first name attribute and high if the data changes frequently like e-mails. Timeliness represents data suitability for a certain problem at a fixed moment. Consistency represents the degree of violations of the predefined semantic rules. Nowadays, data can be qualified by Volume, Variety, Veracity, Velocity, and Value.

2.5.2 Four pillar for data quality management

In order to manage data quality in an organized way, four principal steps have to be followed. These steps were named by Jonathan as the four pillars for data quality management [Geiger 2004].

The first step is data profiling, the purpose of it is to gain an idea about the stored data and its quality compared to the quality specifications. Data profiling in the data warehouse world is called "sources system analysis". It allows the quality manager to have an idea about the accuracy and the completeness of the stored data. Data profiling helps also in detecting duplicate values in the case of having too much data than expected.

Once data profiling is done, the quality manager can have an idea about the quality problems in the stored data. Four solutions are possible to deal with the affected data: (1) Remove the data in the case where the problem is hard to deal with. (2) Accept the data where the error in the data can be ignored. (3) Correction of the data. For example, in the case of the duplicate values, we can select one tuple and delete the others and finally (4) inserting a default value, especially in the case of missing values.

Data integration is also an important process to achieve a high data quality. Specially, when the data is gathered from multiple sources, the same real world entity can be represented differently at the level of each source. Doing that, helps deleting duplicate records which improves the quality of the stored data.

Data augmentation also have an important role in data quality management. It develops meaningful information about clients by the integration of external data from third parties with the stored data.

2.6 Record Linkage

As mentioned in the introduction, in the field of Data Quality, the process of identifying the records that represent the same real-world entity and choosing which one to preserve or how to merge them, is known as the Record Linkage process. From the literature, the RL process can be defined by a three-step process [Christen 2011] as it's depicted in Figure 2.8.

The first step is data standardization. In fact, it has been proven in previous research [Clark 2004] that leaving datasets without standardization of attributes, and detecting schema anomalies can lead to the wrong con-

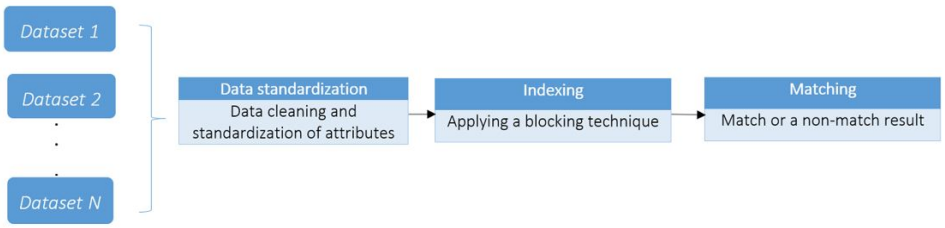

Figure 2.8 – *Main steps of the Record Linkage Process.*

clusion and even merge the wrong tuples. For example, the attribute that represents the name of a person can appear in one dataset as Full Name and in another one as two attributes (First Name & Last Name). So, data standardization is an important step in the RL process.

The second step in the RL process is indexing, the goal of this step is to identify the tuples that will be compared to each other in the matching step. The perfect way to do that in terms of accuracy is the naive approach by comparing each record to all the others. But, of course, that could end up by an unacceptable number of comparisons. For example, comparing two databases with 2 million records each can end up with 4^{12} comparisons and most of these comparisons will end-up with a non-match result. Consequently, indexing aims to reduce the number of comparisons. The most used technique for Indexing in the RL community is known as "BLOCKING".

Blocking is the process that divides the dataset into a set of blocks. All the tuples that are assigned to the same block share a common value known as a Blocking Key-Value (BKV). A blocking key can be chosen as a single attribute. For example, all the records that share the same value for the attribute address are assigned to the same block. Otherwise, a blocking key can also be chosen with the concatenation of several attributes like the first four characters of the First Name and the Zip Code from the address attribute. The decision of selecting which attribute or group of attributes will be used as a blocking key is very important because the blocks that will be created depend on that decision. So, choosing the least error-prone attribute as BK is very important and sometimes we need the intervention of an expert for that decision. Once the indexing step is made, only the records in the same block are compared to each other. Several

blocking techniques were proposed in the literature and will be reviewed in Chapter 3.

The last step on the RL process is matching the indexed records that are in the same block. Matching can be done using a set of string similarity functions that exist in the literature [Levenshtein 1966] or by using a machine-learning algorithm to classify the record as match or non-match [Ouhab *and al.* 2017].

2.6.1 Record Linkage and Big Data challenges

As most of the data cleaning tasks, the arrival of the Big Data era puts new challenges for the Record Linkage process. Most of the traditional blocking techniques are unable to handle the huge amount of data generated every day. Each of the Vs mentioned in the definition of Big Data in the previous sections, brings new challenges that have to be handled by the RL process.

The Volume aspect of Big Data made the Record Linkage process more complex. With more data comes more comparisons, which makes the blocking approaches more important to reduce as much as possible the number of comparisons. The veracity of Big Data makes the first step of the RL process more crucial. Data is collected from various sources with different schemas. Standardizing this data is very important to get good results. Applying the RL process over Big Dataset can end-up with a high computational time, which is not acceptable in the case of BD problems, in which, time is a very important axis. Improving the traditional RL approaches and making them more scalable is a very important to deal with the Velocity of BD. Nowadays, data is collected from different sources without considering if it has commercial value or not. Since Big Data collection is not supervised by specialists (scientist investigators and agencies), selecting the most valuable data for analysis is also a very important step.

2.6.2 Applications domains

Over the past decades, record linkage has been used in various domains for several reasons. RL can be integrated into any process that needs to integrate data that comes from different sources in which duplicate values must be removed in order to extract correct conclusions out of the collected data. RL has been used in domains such as medical care, analyzing census data, crime and fraud detection, and privacy-preserving.

In the USA, annually, the National Center for Health Statistics asks a set of selected hospitals to submit a list of all the discharges patients and all the visits in the outpatient department. Their goal is to link the data collected from hospitals with other collected data such as the Nation Death Index in order to get information about the mortality in hospital after discharge, to do so, they use the RL process. Another example is linking patients' data from different services in order to improve the quality of services in hospitals.

Record Linkage is also being used by the online publishers in order to detect duplicates publications in their storage and also to detect plagiarism in the documents stored in their databases. Business companies also take advantage of the RL process by detecting and removing duplicates entities from their databases in order to better address their advertisement investments and to reduce the dedicated budget.

2.7 Conclusion

Bad data costs trillions of dollars per year in the US alone. The reason of these losses is that various processes such as decision making accommodate these bad data every day. Organizations around the world are now more aware about the importance of data quality in their databases, where a lot of money is invested in order to improve the quality of the stored data.

One of the main important processes in the field of data quality is the Record Linkage (RL) process. The goal of record linkage is to identify

the tuples that refer to the same real world entity and merge them into a single one. RL helps improving data quality significantly by removing all the records that refers to the same real-word entity.

In this chapter, we presented the preliminaries that have to be known in order to better understand the problem tackled in this thesis in a general way. Moreover, this chapter presents some technologies and architecture such as Map-Reduce and Hadoop that will be used later in our contribution. In the next chapter, the current state of art of RL will be presented.

State of the art

Sommaire

3.1 Introduction

In the previous chapter, we saw the importance of data quality and its huge impact on all the levels of data analyzing and on the decision support systems. Detecting duplicates values that represent the same real-word entity is one of the most common data quality problems that have to be taking into consideration. Several communities have worked on the Record Linkage process in the past years including the databases community and the Artificial Intelligence community. Many approaches and techniques have been proposed on all the levels of the RL process (indexing, blocking, and matching). This section covers most of the works that exist in the literature concerning the RL process and all its aspects.

3.2 Indexing/blocking

Indexing is the second step of the RL process (after data standardizing). It is considered as the most important step in Record Linkage. The goal of indexing and blocking is to reduce the comparison space compared to the Cartesian product.

3.2.1 Standard blocking

The most used technique for indexing is the blocking technique. It has been used in several RL approaches for many years. The first proposed blocking technique is the standard blocking [Fellegi & Sunter 1969]. The idea behind it, is to regroup all the tuples that share a common Blocking Key-Value into the same block. A Blocking Key can be selected as a single attribute or a concatenation of multiple ones. Doing that, only the tuples that are in the same block can be compared to each other. The major disadvantage of the standard blocking is that generating a BKV based on a possible erroneous attribute (like names and addresses) can lead to a wrong match. To avoid this problem, the least error-prone attributes have to be selected [Gu & Baxter 2004] or by using multiple blocking keys.

Figure 3.1 shows a demonstration of the standard blocking. In this example, the city attribute is used as the blocking key. As we can see, all

the tuples that have the same city are grouped into the same block. Doing that helps to avoid comparing all the tuples of the dataset to each other and comparing only the records of the same block.

Using the city attribute as a blocking key helped to reduce the comparison number from 10 matching operations to only 2 operations. But, using this blocking key will not detect R4 as a true match with R2 and R5 since they are not into the same block. A possible solution is to use a blocking key formed by the concatenation of several parts of different attributes like the first three characters from the First Name concatenated with the first character from the city. As a result, two blocks are obtained. Block1 (BenS) groups records R1 and R2, and Block2 (BerS) groups records R3, R4 and R5.

ID	First Name	Last Name	City	Age
R1	Ben khaled	Hamid	Saida	27
R2	Benkhaled	Hamid Naceur	Saida	1993
R3	Berrabah	Djamal	Sidi Bel Abbes	1976
R4	Be rabbah	Djamel	SBA	1976
R5	Berrbah	Djamil	Sidi Bel abbes	44

Blocks (blocking key)	Records
Block 1 (Saida)	R1,R2
Block 2 (Sidi Bel Abbes)	R3,R5
Block 3 (SBA)	R4

Figure 3.1 – *Standard blocking example with the address as a blocking key*

3.2.2 Q-gram indexing

Another powerful blocking technique is Q-gram indexing [Gravano *and al.* 2001]. In this approach, the blocking key-value gets divided into a set of sub-strings of a size Q, then a number of these sub-string get concatenated to form a new blocking key value. All the obtained new strings will be used as blocking keys. In that way, the same record can get assigned to different blocks. This approach was improved

later in [Elziky *and al.* 2018] by converting the blocking keys into digits using the ASCII code.

The results that have been published in the survey of [Christen 2011] showed that the Q-gram indexing technique is less performing compared to the standard blocking regarding the efficiency when the attributes are of good quality. In addition to that, Q-gram blocking takes more execution time because of the recursive generation of the sub-strings. However, this approach gives the advantage of avoiding the impacts of the erroneous attributes which gives better matching results.

3.2.3 Suffix array-based indexing

The suffix array-based indexing [Aizawa & Oyama 2005] is another indexing technique that exists in the literature. It consists of generating a set of suffixes of a minimum length chosen by the user from each blocking key. Once that is done, all the generated suffixes will be used as a new blocking keys, which allows the insertion of the same record into different block same as the Q-Gram indexing. This approach has been extended in [De Vries *and al.* 2009]. The authors proposed the idea of merging two blocks if the similarity between their Id-Suffixes reaches a fixed threshold.

3.2.4 Sorted neighborhood

In [Hernández & Stolfo 1995], the authors proposed another indexing approach named the sorted neighborhood approach. The first step in this approach is to generate the blocking keys for all the attributes then sort them alphabetically according to their BKV. Once that is done, a sliding window with as size W is moved over the records. At each step, the records that are in the window covering range are compared to each other. Using a sliding window can reduce the number of comparisons for each record to (2W - 1) [Gu & Baxter 2004]. The main limitation of this approach is that when the size of the sliding window is too small compared to the size of the database, a large number of records with the same BKV will never be in the range of the same window and will not be compared to each other. This approach was improved later in [Christen 2007]

[Yan *and al.* 2007] by using a dynamic changing window and an inverted indexing array.

Figure 3.2 shows an illustration of the sorted neighborhood approach. The same records of Figure 3.1 are used in this example. The window size in this example is w=3. The final candidate pairs list is: {(R1,R2), (R1,R4), (R2, R4), (R2,R3), (R3,R4), (R4,R5), (R3,R5)}.

Window position	Blocking key values (City)	ID
1	Saida	R1
2	Saida	R2
3	SBA	R4
4	Sidi Bel Abbes	R3
5	Sidi Bel Abbes	R5

Windows range	Pairs
1-3	(R1,R2), (R1,R4), (R2,R4)
2-4	(R2,R4), (R2,R3), (R4,R3)
3-5	(R4,R3), (R4,R5), (R3,R5)

Figure 3.2 – *Sorted neighborhood example with a window size of w =3.*

3.3 Matching

As mentioned above, the last step in the record linkage process is matching the indexed records and deciding whether a two compared pairs represent the same real-world entity or no. Generally, the matching value is normalized between the range of [0,1] where 1 represents an exact match and 0 a total non-match. In this section, a brief overview is provided about the existing matching techniques in the literature.

From the literature, two families of matching techniques exist. The first one is the phonetic encoding. The idea of this technique is to transform a string to a code that represents how the string is pronounced. A variety of phonetic encoding algorithms exists in the literature (Soundex and phonex, phoenix, NYSIIS and Double-Metaphone). The second matching technique family is the pattern finding. The main idea of these techniques is to measure the similarity between two words without any transformation using a set of string similarity metrics such as the edit distance.

3.3.1 Phonetic encoding

a- **Soundex**

Soundex is considered as one of the most efficient phonetic encoding functions. It transforms Strings to the way they are pronounced so they can be compared to each other without taking into account the spelling errors. Using Soundex, names like ALLAN and ALLEN are both represented with the same code "A450" which makes it easy to match between the two names. The main steps of Soundex are:

— Keep the first letter of the String.
— Replace all the consonants using the following rules: (0 for the characters A, E, H, I, O, U, W, Y. 1 for the characters B, F, P, V. 2 for C, G, J, K, Q, S, X, Z. 3 for D, T. 4 for L and 5 replaces M, N. 6 replaces the character R.
— In the case where the string is too short, the algorithm completes the three numbers after the first character by zeros.

b- **NYSIIS (New York State Identification Intelligence System)**

NYSIIS has the same idea and purpose as the Soundex algorithm. The difference is that NYSIIS returns a code composed of letters which is not the case of Soundex. The NYSIIS algorithm increases the accuracy by 2.7% compared to Soundex [Rajkovic & Jankovic 2007]. The basic rules of the NYSIIS algorithm are the transformation of the first characters where: (MAC is replaced by MCC and KN become NN, K to C, PH-PF to FF, SCH to SSS) and the last characters (EE-IE to Y, DT-RT-RD-NT-ND to D).

3.3.2 Pattern finding

The second family of matching techniques is pattern finding. A variety of matching algorithm exists in the literature.

a- **Edit distance**

Edit distance is also known as the Levenshtein distance. It was proposed in 1965 by Vladimir Levenshtein. It is considered as one of the most used metrics in order to measure the similarity between two strings. Generally, it is defined as the number of inserts, deletes and updates in order to transform one string to another. To better understand it, the following example shows a demonstration of how to calculate the costs of passing from a word to another.

Table 3.1 – *Edit distance example*

Word 1	**Word 2**	**Operation**	**Cost**
I		Delete (I)	1
N	E	Substitute (E)	1
T	X	Substitute (X)	1
E	E	Comparison	0
	C	Insert (C)	1
N	U	Substitute (U)	1
T	T	Comparison	0
I	I	Comparison	0
O	O	Comparison	0
N	N	Comparison	0
		Sum	5

From the above example, we can see that the edit distance between the two words (Intention, Execution) is the sum of the costs to transform the first string into the second one which is equal to Five. The steps that are depicted in the example are not the only solution to transform the first string into the second one but are the ones with the least cost.

b- **Jaro-Winkler**

The Jaro-Winkler is a string similarity metric that was proposed by William E. Winkler in 1990 as an extension to Jaro distance. In order to measure the Jaro-Winkler similarity between two strings, the first step is to measure the tradition Jaro similarity which is defined as:

$$Jaro\_Sim(s_1, s_2) = \begin{cases} 0, & m = 0 \\ \frac{1}{3}(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}), & otherwise \end{cases} \quad (3.1)$$

Where:

— s represents the length of the string.

— m represents the number of the common characters between the compared sequences with the same index.

— t represents the half number of transpositions.

In order to improve the previous metric, William E. Winkler uses a prefix scale P in order to favorites the strings that starts with the same prefix L for a maximum length of Four. The Jaro-Winkler similarity is defined as follows:

$$JaroWinlker\_Sim(s_1, s_2) = Jaro\_Sim(s_1, s_2) + LP(1 - Jaro\_Sim(s_1, s_2)) \quad (3.2)$$

Where:

— $Jaro\_Sim(s_1, s_2)$ is the Jaro similarity between the strings.

— L is the prefix length.

— P is a scaling factor (a constant that generally takes the value 0.1).

c- **Jaccard distance**

The Jaccard distance is usually used to measure the similarity between two sample sets which can be the case of Strings. In order measure the Jaccard distance, first the Jaccard coefficient has to be calculated which is defined as :

$$Jaccard(A, B) = \frac{A \cap B}{A \cup B} \quad (3.3)$$

Once that is done, the Jaccard distance is obtained only by the subtraction of the Jaccard coefficient from 1.

$$Jaccard_{distance}(A, B) = 1 - Jaccard(A, B) \quad (3.4)$$

3.4 Record Linkage with Map-Reduce

The Map-Reduce paradigm fits perfectly with the record linkage based on the standard blocking approach. The blocking step can be implemented easily as a mapping step while the matching can be done as the reducing step (See Figure 3.3).

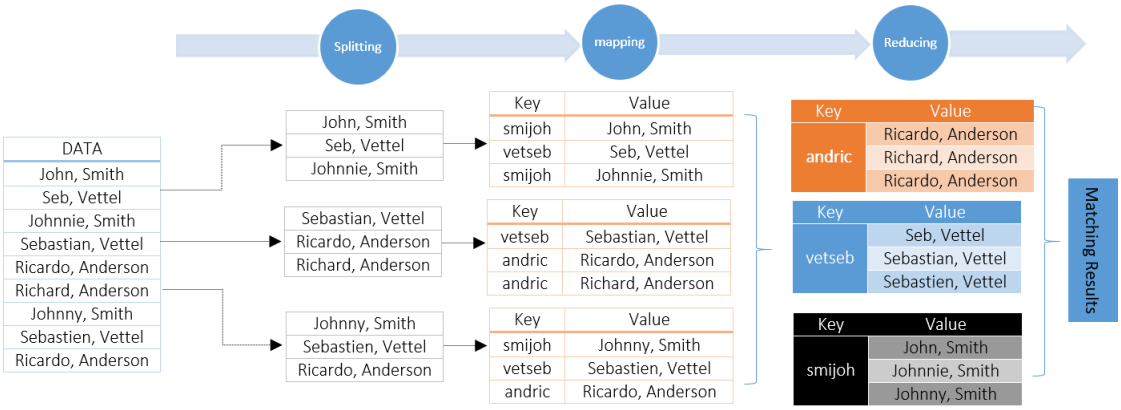

Figure 3.3 – *Record Linkage with Map-Reduce*

3.5 Indexing based on Map-Reduce

With the arrival of the Big Data era, most of the blocking approaches discussed in section 3.2 have become unable to handle large datasets and take a lot of executing time. Recently, the RL community started proposing solutions by adapting the traditional approaches to the new BD requirements and challenges. In this section, we make a brief overview of the most important approaches that were proposed about RL for Big Data.

[Kolb *and al.* 2012] proposed a Map-Reduce based Sorted Neighborhood indexing approach . The authors mentioned two main challenges that have to be solved to implement the SN method in a Map-Reduce paradigm. The first one is the ordering problem. In the traditional SN, once all the blocking keys are generated for each entity, all the entities in the dataset get sorted according to the generated blocking key. As a result, the Map function has to ensure that all the entities assigned to the reducer R_i have a smaller blocking key than the entities directed to the reducer $R_{(i+1)}$. The second main problem for implementing SN in a

distributed architecture is that the SN algorithm does not compare only the entities with the same blocking keys but even entities with different blocking keys in the same window range, which makes it difficult to compare the entities that are in the same window range but assigned to different reducers (Boundary entities).

The first challenge was solved by proposing a Map-Reduce job named SRP (Sorted Reduce Partitions). SRP uses a user-defined function to add a part number "i" as a prefix to the blocking key where $1 \leq I \leq R$ (number of reducers). For example, in the case of R = 2, all the entities with a blocking key smaller than 5 get associated with a prefix "1" so they get assigned to the first reducer. All the entities with a higher blocking key get assigned to the second reducer. In that way, SRP makes sure that the sorting problem is solved during data redistribution. The authors solve the second challenge by proposing two different solutions, the first one uses a new MapReduce job named JobSN which gets executed after SRP. JobSN receives as an input the boundary entities from the SRP which identify the boundary entities during the reduce phase. The second solution is RepSN, it is considered as an extension to SRP in which the map function identifies the boundary entities of each part and replicates them in the other part where the window range will move to. In that way, RepSN realizes the SN approach using only one MapReduce Job.

In [Kolb & Rahm 2013], a new Record Linkage framework (Dedoop) was proposed. The authors based their framework on the Map-Reduce paradigm, where the user can select different RL strategies to execute. As an example, the user can choose to evaluate the Cartesian product of all the entities in the dataset or select one of the implemented blocking techniques. Two blocking techniques were implemented in Dedoop (Standard Blocking and Sorted neighborhood). For the standard blocking, the blocks get created during the mapping phase by creating for each entity a < key, value> pair, in which, the key represents the blocking key and the value represents the record. When the mapping is done, all the pairs with the

same key are directed to the same reducers where the similarity computation is calculated. The output of reducing is the result of matching. The sorted neighborhood approach is also implemented in this framework. For matching, the user can select between many string similarity functions that have been implemented in the framework or using a machine learning algorithm as a classifier. Three machine learning algorithms were implemented in Dedoop using the WEKA library (SVM, Decision Tree, Logistic regression).

The authors conducted their experiments on the Google scholar dataset, where 20 record linkage strategies were investigated. For example, they tested the traditional blocking with a single pass strategy and a multiple pass strategy and varying the selected similarity function. The execution time is reduced from around 40 minutes when using the Cartesian product to 3 minutes with a blocking strategy but at the same time, we lose some f-measure since some true matches can be assigned to different blocks.

The authors in [Hsueh *and al.* 2014] proposed a new Record Linkage approach based on the Map-Reduce framework. The proposed approach is composed of two principal steps. The first one is a combination based blocking to deal with the case of entities with multiple blocking keys. The second one is load-balanced matching to solve the problem of skewed keys distribution in the reducing phase where matching between the entities has to be done. For each step, a Map-Reduce job is introduced. The first job is for the combination based blocking, where the mapper generates all the kc combinations from the received blocking keys (kc is the number of blocking keys) and output each combination as a <key,value> pair where the key is the blocking keys combination and the value is the ID of the processed entity. Once the mapping is done, the reduce function is executed to select only the entities that have at least kc common keys to be matched in the next Map-Reduce job. The output of the first job is used as an input to the second one, where each record pairs is associated with a part number which will be used to evenly distribute the similar-

ity computation among all the reducers. For example, all the record pairs with an odd part number are directed to the same reducers while the rest of the record pairs are directed to the other reducer (in the case of 2 reducers). The standard blocking approach was also implemented in [Karapiperis & Verykios 2015] where they proposed a solution for distribution similarity computation evenly between the reducers.

3.6 Automatic blocking keys selection

Several blocking approaches were proposed by the RL community. Each one of them depends in one way or another on good blocking key choice.

Knowing that most of the blocking approaches depend on the initial blocking key selection, proposing an automatic blocking key selection solution is one of the most important priorities of the RL community. In recent years, the Record Linkage community proposed several automatic blocking keys selection approaches. Some of these approaches require the existence of a gold standard dataset since they are based on supervised learning algorithms like ([Vogel & Naumann 2012], [Bilenko *and al.* 2006], [Michelson & Knoblock 2006]).

Vogel and Naumann proposed an automatic approach for the blocking keys selection based on the uni-gram combinations as generated blocking keys [Vogel & Naumann 2012]. The first step in this approach is to generate all the possible uni-gram combinations. For each combination, a duplicate detection algorithm is executed on a reference dataset. After each run, if the number of the detected duplicates is acceptable then the overall blocking key quality is calculated. All the selected blocking keys from this step will be stored and sorted according to their quality. Once the first step is done, the record linkage process is executed on a test dataset using the sorted blocking keys list from the previous step, the best BK that satisfied the chosen stopping criteria is selected.

Bilenko et al proposed to generate a set of blocking predi-

cates. Each predicate can be specified for an attribute in the dataset bilenko2006adaptive. By that way, the problem of the automatic blocking key selection consists of how to select the best subset of blocking predicates that detects the most possible duplicates in the dataset. Authors used the red-blue set cover problem to select the best predicates where the top and the bottom rows are for the positive and negative pairs while the middle row represents all the generated blocking predicates.

Michelson and Knoblock proposed a modified Sequential Covering Algorithm (SCA) to generate the best blocking schemes [Michelson & Knoblock 2006]. They used the matches of the dataset as the positive example of the SCA algorithm. All the previous approaches are based on the existence of a gold standard dataset which is not the case of the real-world datasets.

On the other hand, we have a set of proposed approaches that do not require the existence of gold standard dataset.

For instance, [Kejriwal & Miranker 2013] proposed an unsupervised approach for the blocking key selection. They used the fisher discrimination criterion to select the best blocking keys after the generation of a weakly labeled dataset . Indeed, they treated the situation as a feature selection problem.

[Ramadan & Christen 2015] proposed another unsupervised approach for automatic blocking key selection based on three principal criteria. The first one is the key coverage. It is the number of record pairs covered by each blocking key. The second one is block size, which is the number of records in the same block where they fixed a maximum block size since they treat the problem of real-time Entity Resolution. The last one is the distribution of blocks that represents the variance. The obtained results showed that their approach can return good quality blocking keys in a reasonable time for real-time RL.

Shao and Wang proposed an active blocking scheme learning ap-

proach. They used active learning techniques to select the best blocking scheme [Shao & Wang 2018]. Their approach is composed of two principal steps. The first one is active sampling. During this phase, a balance rate is calculated for each blocking scheme on a given set. The goal is to minimize the balance rate. The second step is active branching. It's where a locally blocking scheme gets searched over the training set. The experiments on four real-world datasets show good results, specifically, concerning Reduction Ratio.

So, we can deduce, from the approaches discussed above, that the problem of blocking keys selection can be considered as a feature selection problem where all possible blocking keys are assumed to be generated and the objective is to select the best subset of these keys that can accelerate the performance of the RL approach. Feature selection is considered as an NP-Hard problem. The use of exact methods risks presenting a high computational complexity. Consequently, in machine learning, the problem of feature selection is solved using stochastic methods such as heuristics and meta-heuristics. There are three feature selection strategies: Filter, Hybrid and embedded [Chandrashekar & Sahin 2014].

In approaches using the filter, the features are sorted according to ranking criteria where all the features that have a score below a defined threshold are eliminated. Several ranking methods were proposed in the literature like correlation criteria [Guyon & Elisseeff 2003] and the conditional mutual information-based criteria [Fleuret 2004]. However, the wrapper methods are considered more efficient than the filters since they are based on searching algorithm guided by a classifier to measure the objective function. Most of the existing approaches using the wrapper are based on meta-heuristics as the searching algorithm since meta-heuristics are suitable for solving NP-Hard problems.

Several meta-heuristic algorithms have been used to solve the problem of feature selection. Some of them used directly one meta-heuristic and others tried to combine two or more meta-heuristics to improve both the

exploration and the exploitation sides.

Genetic algorithms (GA) are among the first meta-heuristics used to solve the problem of attribute selection [Bala *and al.* 1995]. Indeed, the features can be used as the initial population for GA, in such a way that each subset of features is composed of binary values (1 or 0). Value 1 means that the ith feature is included in the subset of features and 0 means that the feature is not included. To measure the fitness of a current subset of features, the ID3 algorithm is used to compute the accuracy of classification that was used as the fitness function for the GA algorithm. The obtained results showed that combining the GA and ID3 algorithms gives better results than the traditional attributes ranking approaches.

The Firefly algorithm (FFA) is another meta-heuristic used in the field of attributes selection [Emary *and al.* 2015]. In this approach, the FFA is used for selecting the best attributes and the K-nearest neighbor is used as a classifier to minimize a given fitness function. This approach was then improved by proposing a recursive FFA where the best-selected features at the first FFA run, form a new dataset including only these best-selected features [Dif & Elberrichi 2019]. Then, the FFA for feature selection is executed on the new dataset to search for a better solution. The obtained results showed an improvement in terms of accuracy.

Other approaches combined two or more meta-heuristics to improve both the exploration and the exploitation sides. [Jona & Nagaveni 2014] proposed a hybrid solution using the Ant colony optimization (ACO) algorithm and Cuckoo Search (CS) algorithm for the attribute selection.

[Dif *and al.* 2018] tried three different hybrid meta-heuristics approaches to see how hybridization could improve the feature selection. In the first one, the authors introduced the crossover function of the GA algorithm into the Penguins Search Optimization Algorithm (PeSOA). The goal was to avoid the rapid convergence of PeSOA. The second hybrid

approach was a combination of the Firefly (FA) and Deferential Evolution (DE) algorithms [Zhang *and al.* 2016]. The goal was to use the advantage of the local search given by the DE algorithm. The third test of hybrid approach was by introducing the mutation and cloning functions of the Artificial Immune System to create new Bats for the BAT algorithm initialization phase. The obtained results of these works showed that using a single meta-heuristic performs better than the hybrid approaches, specifically regarding the high complexity of the hybridization approaches. The last methods are the embedded methods that combine Filters and Wrappers [Maldonado *and al.* 2011].

3.7 Record Linkage For Arabic Language

In the literature, several RL approaches and frameworks were proposed but most of them do not take into account the Arabic characters. In this section, an overview is given about the existed RL approaches in the literature dedicated to the Arabic language. To the best of our knowledge, only three works address this problem in a particular way.

[Alian *and al.* 2017] proposed an extension to the DySimII approach, the idea was to add an additional preprocessing step before indexing. During this step, the Arabic words are transcribed into English using the Buckwalter system which transforms every Arabic letter into its corresponding one in the English language. The authors chose to use existed similarity functions to match between the indexed records where all the attributes are concerned during this step. Their experiments were conducted on an artificial Arabic dataset in which they introduced three types of errors (variations corrupter, edit value corrupter, and keyboard corrupter). The obtained results were acceptable where accuracy reaches 71%. Compared to some stemming techniques, translation performs better when the percentage of corrupted attributes is high.

[Higazy *and al.* 2013] proposed an RL approach that includes a set of cleaning functions specified for the Arabic language [Higazy *and al.* 2013].

These functions were implemented as an additional step in the data standardizing step. The authors equipped their framework by a number of built-in rules such as the one that transforms all the characters (آ ، إ ، أ ، ٱ) to their equivalent (ا). The proposed framework also allows the user to define a set of standardization rules for the String level like removing titles and prefixes like (Mr., Dr...). As blocking function, they chose to do two steps blocking. In the first step, standard blocking is applied over the data. Once the first blocks are created, the sorted neighborhood indexing is applied at the level of each block which creates new sub-blocks from each one. The experiments show that the implemented function for the Arabic adjustments improves the obtained results were the detected duplicates on an Arabic data set increased from 35 to 54.

[Ouhab *and al.* 2017] also proposed an RL approach that supports data in Arabic and English languages. The proposed framework allows the user to turn on or off a specified preprocessing step dedicated to the Arabic language. In this step, predefined rules such as the ones in [Higazy *and al.* 2013] are applied to standardize the Arabic data. Such as the character ي is replaced by the character ى, and the character ؤ is replaced by its equivalent in the Arabic language و.

In addition to the approaches discussed above, other works have been done to support RL with the Arabic language. Most of them are an extension to already existed similarity metrics. For example, a variant of the Levenshtein distance to support Arabic characters was proposed in [Hicham 2012]. El-Shishtawy also proposed an extension to the edit distance in which tokens are treated as individual characters [El-Shishtawy 2013]. An Arabic version of the Soundex algorithm was proposed in [Aqeel *and al.* 2006].

3.8 Block sizes control

In this section, we show a brief overview of the existed methods for controlling the size of the generated RL blocks. To the best of our

knowledge, only two works addressed this problem in a particular way [Fisher *and al.* 2015] [Nascimento *and al.* 2020].

[Fisher *and al.* 2015] proposed a clustering approach to control the size of the generated blocks. The main idea of their proposed approach is to divide the blocks that have a large size using another blocking key. Then, merge all the blocks with small sizes and similar blocking keys values. To achieve their goal, they proposed two different methods. The first one aims to decrease block similarity and the second one to increase block sizes. In addition to that, they proposed a penalty function, which allows merging two similar blocks even if their combined size is larger than the predefined maximum threshold. Doing that, increases the block quality. They conducted their experiments on three real-world datasets, the obtained results showed that their approach gives better results than three blocking approaches (Standard Blocking, Sorted Neighborhood, and Soundex encoding).

[Nascimento *and al.* 2020] proposed another approach to control the size of blocks, they proposed a method composed of 4 principal steps. The first step is Shrinking Blocks, in which, many entities get removed from each block. The removed entities are the ones with the lowest mean co-occurrence score value in each block. The goal of this step is to increase Reduction Ration. The second step named block splitting, in which, all the blocks that have a size larger than a predefined value gets split based on the co-occurrence of their entities. The third step is for merging the small blocks, where they proposed to merge the blocks that share the maximum number of entities after their intersection. The last step is the blocks excluding, where the authors proposed to remove the block with the lowest block co-occurrence score.

[Das Sarma *and al.* 2012] did not address the problem of block sizes control directly in their proposed approach for the automatic blocking schema generation. However, they proposed a post-processing step aim to

merge all the generated small canopies under the name of Rolling up small canopies. Their aim was to increase the overall recall of their approach by bringing together more possible duplicated entities.

3.9 Comparative study

In this section, a comparison study is provided about the state of art approaches of Record Linkage. This comparison is based on 5 principal points: (1) Type of indexation. (2) Arabic language support. (3) Block sizes control (whether the approach provides a mechanism to control the size of the generated blocks or not). (4) Big Data support (Whether the proposed approach is adapted to the Big Data challenges or not) and finally, (5) Automatic blocking keys selection which means if the proposed approach needs the intervention of an expert to select the blocking keys manually or the task is automatized.

Table 3.2 provides a comparative study of the approaches discussed in section Chapter 3. From the table we can extract the following conclusions:

Indexation

Most of the existed RL approaches use the standard blocking and the sorted neighborhood as an indexing/blocking step. These two indexing techniques proved their efficiency over the other blocking techniques in most of the state of art approaches. Moreover, works such as ([Higazy *and al.* 2013], [Kolb & Rahm 2013]) even tried to combine the standard blocking with the sorted neighborhood indexing as a two-step process to improve the quality of the generated blocks. However, other blocking techniques such as the one used is [Aizawa & Oyama 2005] show better results when dealing with datasets of bad quality with erroneous attributes. Even the proposed works that take into account the new Big Data challenges such as ([Kolb *and al.* 2012], [Hsueh *and al.* 2014], [Karapiperis & Verykios 2015]) chose to use the sorted neighborhood and the standard blocking with a parallel implementation over the other blocking techniques.

Table 3.2 – *Comparative study (state of art approaches)*

Approach	Indexation	Arabic language support	Block size control	Big Data support	Automatic blocking keys selection
[Fellegi & Sunter 1969]	Standard blocking				
[Hernández & Stolfo 1995]	Sorted neighborhood				
[McCallum *and al.* 2000]	Canopy clustering				
[Gravano *and al.* 2001]	Q-gram indexing				
[Aizawa & Oyama 2005]	Suffix-array based indexing				
[Bilenko *and al.* 2006]	-				X
[Michelson & Knoblock 2006]	-				X
[Vogel & Naumann 2012]	Sorted neighborhood				X
[Kolb *and al.* 2012]	Sorted neighborhood			X	
[Kolb & Rahm 2013]	Standard blocking / Sorted neighborhood			X	
[Kejriwal & Miranker 2013]	-				X
[Higazy *and al.* 2013]	Standard blocking / Sorted neighborhood	X			
[Hsueh *and al.* 2014]	Standard blocking			X	
[Karapiperis & Verykios 2015]5	Standard blocking			X	
[Ramadan & Christen 2015]	Sorted neighborhood				X
[Fisher *and al.* 2015]	-		X		
[Ouhab *and al.* 2017]	Token blocking	X			X
[Alian *and al.* 2017]	Standard blocking	X			
[Shao & Wang 2018]	-				X
[Nascimento *and al.* 2020]	-		X		

Arabic language support

To the best of our knowledge, only three works treated the problem of RL for the Arabic language. Most of the other works do not include non-Latin characters in their frameworks. Alian and al chose to add an additional transliteration step to deal with Arabic characters [Alian *and al.* 2017], while Higazy and al chose to use a set of built-in rules to transform Arabic characters into standardized ones [Higazy *and al.* 2013]. However, both of these works are based on the use of transliteration which transforms Arabic letters to their correspondence ones in the Latin alphabet. Several transliteration systems for the Arabic language exist, but none of them proved itself as a standard one which makes these approaches under discussion on which transliteration system to use.

Block sizes control

All the proposed blocking techniques in ([Fellegi & Sunter 1969], [Hernández & Stolfo 1995], [McCallum *and al.* 2000], [Gravano *and al.* 2001], [Aizawa & Oyama 2005]) do not have a mechanism to control the size of the generated blocks which can be an impediment in the face of real-time RL or privacy preserving RL. Only two works tried to propose solutions to this problem ([Fisher *and al.* 2015], [Nascimento *and al.* 2020]).

Big Data support

As mentioned in the previous chapters, the arrival of Big Data made the traditional indexing/blocking approaches obsolete and cannot support the huge amount of data nowadays. Several approache[Kolb *and al.* 2012],[Kolb & Rahm 2013], [Hsueh *and al.* 2014], [Karapiperis & Verykios 2015]) tried to adapt the tradition indexing approaches by using cloud and parallel computing architectures such as MapReduce with its correspondent implementation Hadoop. [Kolb *and al.* 2012] proposed an adapted solution for the sorted neighborhood indexing based on Map-Reduce while [Karapiperis & Verykios 2015] proposed a parallel version of standard blocking with a solution

to distribute the similarity computations evenly over the reduces. [Kolb & Rahm 2013] proposed an RL based on MapReduce framework that includes both SN and standard blocking.

Automatic blocking keys selection

([Bilenko *and al.* 2006], [Michelson & Knoblock 2006], [Vogel & Naumann 2012], [Kejriwal & Miranker 2013], [Ramadan & Christen 2015], [Ouhab *and al.* 2017], [Shao & Wang 2018]) proposed automatic blocking keys selection approaches. However, most of these works depend on the existing of labeled data which is not the case of real-world datasets. Moreover, works which generate labeled data automatically such as the work of [Ouhab *and al.* 2017], require an additional step in order to label the training data which means additional execution time.

3.10 Conclusion

The large applications of Record Linkage in various domains made the RL literature a very rich one. Most of the existed works in the literature focus on four principal points: Indexing/blocking, Facing Big Data challenges, Controlling the size of the blocks, and dealing with non-Latin characters. In this chapter, we tried to cover most of the important works that exist to resolve these challenges. Nearly 25 main works were studied, analyzed, and compared based on the following points : (Indexation, Arabic language support, Block sizes control, Big Data support, and the automatic selection of blocking keys).

The comparative study shows the main limitations of the analyzed works. In general, none of the existed works covers all the aforementioned points. Based on this, in the next chapter, we will propose a new blocking technique in the field of RL. The proposed blocking technique is associated with a parallel implementation that helps to deal with the Big Data challenges and a mechanism to control the sizes of the generated blocks. Moreover, we propose a methodology to adapt our solution to the Arabic Language

Contribution

Sommaire

4.1 Introduction

In this chapter, we present our contributions to the field of Record Linkage. A solution is proposed for each of the challenges discussed in the introduction chapter. In section 4.2, we present a new blocking technique based on the K-Modes algorithm as a blocking step and adaptive filtering as a post-processing step to blocking. Moreover, a parallel implementation based on the Map-Reduce paradigm is used in order to deal with the new Big Data challenges. Section 4.3 presents a novel automatic blocking key selection based on meta-heuristic. The proposed approach automatizes the selection of the attributes that create the blocking keys without the need of a domain expert intervention. In section 4.3, a new methodology is proposed that control the sizes of the generated blocks. This methodology, makes the K-Modes based RL suitable for real-time applications such as the problem of real-time RL. Finally, section 4.4 shows how we adapted our proposition to be used in the case of the Arabic language. Each of these propositions is evaluated in the next chapter in order to show their performance and efficiency.

4.2 K-Modes Based RL in a distributed environment

In this section, we present a new scalable approach in the field of duplicates detection. Our approach is based on the K-Modes algorithm as an indexing step and Map-Reduce for the parallel execution. K-Modes clustering divides the data into a set of non-overlapping clusters; each one contains records that may refer to the same real-world entity. Once the clustering is done, the adaptive filtering is executed as a post-processing step. Most of the records that are in the same cluster but do not represent the same entity will be ignored during the matching phase. Finally, all remaining records of the same block are compared to each other using a set of string similarity metrics that already exist in the literature.

Two main Map-Reduce jobs are presented. The first one is IBPK (Indexing based on Parallel K-Modes). IBPK receives the blocks of the dataset

as input and divides them to a set of clusters where all the records that are close to each other in terms of similarity are brought together. This job is explained with details in section 4.2.3. The output of IBPK will be used as an input for the second job named Parallel Matching with Filtering (PMF). PMF executes the matching step in a parallel way detailed in section 4.2.4.

All in all, our proposed approach can be summarized by the diagram depicted in Figure 4.1, in which we have our datasets as input to the first Map-Reduce job. IBPK will bring the most possible matches record together into the same clusters. The output of IBPK will be used as input for the second job where matching is done.

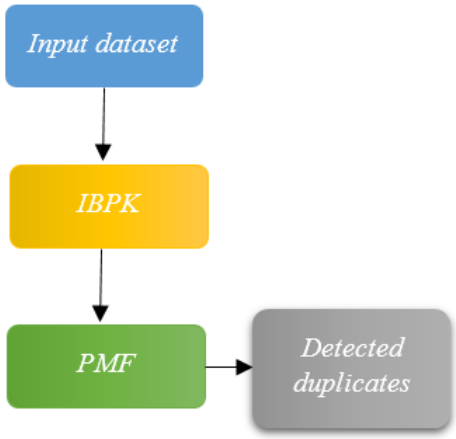

Figure 4.1 – *General approach's steps*

4.2.1 Notation

We present the notation used in the rest of this section. First of all, a dataset is denoted as D. In our work we suppose that the dataset D already exists on a distributed file system and divided into a set of distributed blocks where each block is located on a machine. A dataset D is defined as : $D = \{B_1, B_2, \ldots, B_i\}$ with $(1 \leq i \leq N)$ where N is the number of blocks. Each $B_i \in D$ is composed of a set of tuples and defined as: $B = \{t_1, t_2, \ldots, t_j\}$ with $(1 \leq j \leq |B|)$, where $|B|$ is the size of the block (number of instances in a specified block). Each $t_i \in B$ is composed of a set of

attributes $A = (Att_1, Att_2, \ldots, Att_k)$ with $(1 \leq k \leq |A|)$ where $|A|$ is the number of attributes in the dataset.

4.2.2 Indexing Based On Parallel K-Modes (IBPK)

K-Modes algorithm was first introduced in 1998 by HUANG [Huang 1998]. It was proposed as an extension to the famous classical clustering algorithm K-Means. The main advantage of K-Modes is its ability to deal directly with categorical data which was not the case with K-Means that accepts only numerical features. Using K-Modes instead of K-Means helped us to avoid the data numerical conversion which is an expensive task specifically when dealing with very large datasets such as the case of Big Data.

K-Modes clustering is a 4-step process:

— The first step is to select K initial modes. Two initial modes selection approaches were proposed: (1) Uses the first K distinct categorical objects as initial modes. (2) Computes the frequencies of each category for each attribute and sort them in descending order. Once that is done, the most frequent categories get assigned to the first initial modes.

— For each categorical object, the dissimilarity to all clusters' modes gets computed (using equation 4.1 : distance between two object X and Y). Then, each object gets assigned to its nearest cluster.

$$Distance(X, Y) = \sum_{i=0}^{m} \alpha(X_i, Y_i) \quad \text{With} \quad \alpha = \begin{cases} 0, & X_i = Y_i \\ 1, & X_i \neq Y_i \end{cases} \tag{4.1}$$

— Once all the objects get assigned to their nearest clusters, the modes of each cluster are recomputed using a frequency-based method, then we reallocate each object to its nearest cluster.

— Repeat the previous step until no object changes its assignment.

The authors in [Aranganayagi & Thangavel 2009] proposed another approach to measures the similarity between the categorical objects for K-Modes. It is based on the product of the proportion of attributes value in the cluster and the dataset.

In our work, we use the parallel model of K-Modes that was proposed in [Tao *and al.* 2015], where we use a model on how K-Modes can be adapted to the Map-Reduce paradigm.

The first step in our proposed process IBPK is the blocking keys generation. Since we are considering that the dataset already exists in a distributed file system. Then, for each tuple t_j in a block B_i, one or multiples blocking key get generated. The idea behind blocking keys generation is to use them as the clustering attributes instead of using all the dataset attributes, this can reduce significantly the execution time and maintains good clustering results since the blocking keys contain the most important information about the records. Figure 4.2 shows a passage from the famous Restaurant dataset which is used to test most of the blocking techniques. In this example, the blocking key is formed by the concatenation of the restaurant's city and the phone number.

BK	Name	Address	City	Phone	Type
losangeles310/246-1501	arnie morton's of chicago	435 s. la cienega blv	los angeles	310/246-1501	american
studiocity818/762-1221	art's delicatessen	12224 ventura blvd.	studio city	818/762-1221	american

Figure 4.2 – *Blocking key generation example*

Once the blocking keys generation is done, the parallel K-Modes is executed over the data using only the generated blocking keys for clustering. The parallel K-Modes model is composed of 4 principal steps. Firstly, each record will be represented in the map function as < key, value> pair where the key is the cluster-ID and the value as the record ID.

In the second step, the dissimilarity is measured between each object and the modes of clusters. Each object will be assigned to its new nearest cluster with updating the cluster- ID in the < key, value> pair. Thirdly, all

the records with the same cluster-ID as a key get combined, then compute the new mode of each cluster. Step 2 and 3 are repeated until no object changes its assignment. Finally, the reducer task will write the results as files into the distributed file system as follows: (cluster id, <record>).

IBPK job is summarized in Figure 4.3.

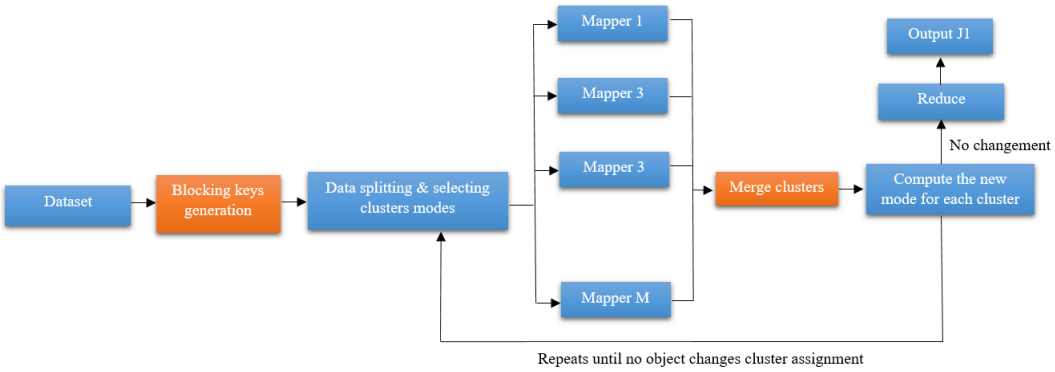

Figure 4.3 – *Indexing based on Parallel K-Modes (IBPK)*

4.2.3 Parallel Matching With Filtering (PMF)

Once the IBPK step is done, its output is used as an input to the second Map-Reduce job. Each Mapper in PMF generates pairs of <key, value > where the key is the Cluster ID and the value is the record or the ID of it. All the pairs with the same cluster-ID are directed to the same reducer. Before starting matching between the records, the adaptive filtering is executed first in order to remove more unlikable matches from the pairwise comparison step. The idea of adapting filtering was inspired by [Gu & Baxter 2004].

Two filtering techniques are used in our approach. The first one is length filtering. The idea of this one is to select one blocking key as a filtering variable for each of the compared records, then the difference between their lengths get measured, if it's bigger than a predefined value, the two records are declared as unlikable matches and will be ignored from the matching step.

The second technique is count filtering. In this technique, the filtering

variables of the compared records get divided into a set of bi-grams. After that, the number of the commons bi-grams (CB) between them get counted and compared to the variable C_{min} which is equal to max(S1,S2) - 2k + 1 where K is the edit distance between the filtering variables. Finally, if $CB > C_{min}$ then the two records are declared as unlikable matches and get ignored from the matching step.

When the adaptive filtering is done, all the records of the same reducer are compared to each other using a set of string similarity metrics such as the Edit distance, Jaro Winkler similarity, and Jaccard similarity.

At this stage in our approach, the user can execute the matching step directly or he can use a similarity based on the Rank Order centroids concept (ROC) ([McCaffrey 2009]). ROC assigns for each attributes a weight calculated directly from a set of ranks entered by the user (First, Second ...) using equation 4.2.

$$W(Att_k) = \frac{\sum_{i=rank(Att_k)}^{n} \frac{1}{i}}{n} \tag{4.2}$$

For example, in the case of the restaurant dataset depicted in Figure 4.2, in which, we have 4 attributes (Name, Address, City, and Phone). If the user entered the ranks as follows (1: Name, 2: City, 3: Address, 4: Phone), then the weights calculated by the ROC concepts are:

$$W(Name) = \frac{\sum_{i=1}^{4} \frac{1}{i}}{4} = \frac{\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}}{4} = 0.52$$

$$W(City) = \frac{\sum_{i=2}^{4} \frac{1}{i}}{4} = \frac{\frac{1}{2} + \frac{1}{3} + \frac{1}{4}}{4} = 0.27$$

$$W(Address) = \frac{\sum_{i=3}^{4} \frac{1}{i}}{4} = \frac{\frac{1}{3} + \frac{1}{4}}{4} = 0.14$$

$$W(phone) = \frac{\sum_{i=4}^{4} \frac{1}{i}}{4} = \frac{\frac{1}{4}}{4} = 0.0625$$

Using these weights, one can easily calculate the similarity score of two records using equation 4.3.

$$Sim(R1, R2) = \frac{SimFunction(R_1Att_i, R_2Att_i) * W(Att_i)}{N} \tag{4.3}$$

Where N is the number of attributes used in matching and SimFunction can be any string similarity function like Jaro Winkler or Jaccard similarity. Finally, the reducer writes the result of matching into the distributed file system.

The Parallel Matching with Filtering is explained with an example in Figure 4.4 where the output of IBPK is used as an input to PMF. The data get split over the mappers in which the map function generates pairs of <key, value > with the key as the cluster-ID and the value as the record. When the mapping is done, all the pairs with the same key are directed to the same reducer. For each reducer, the pairs get filtered to remove the unlikable matches before starting the matching phase. Finally, matching is executed in each reducer generating the final result which is the detected duplicate values.

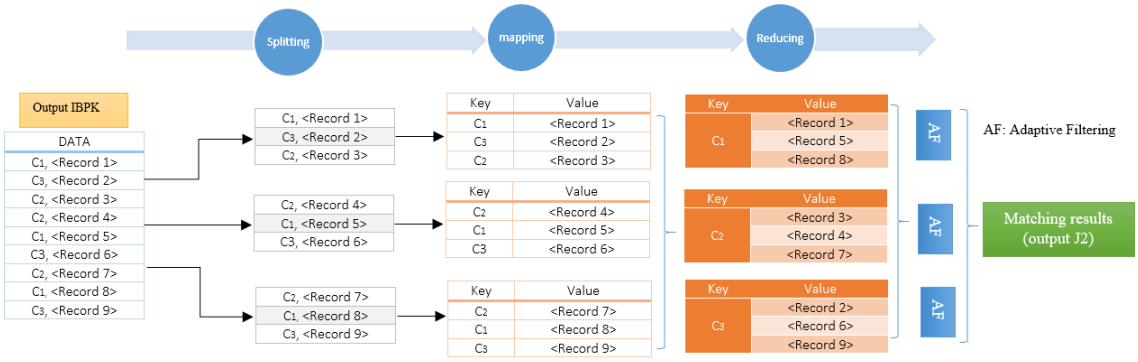

Figure 4.4 – *Parallel Matching with Filtering (PMF)*

4.3 Automatic blocking keys selection

In this section, we propose a novel approach for the automatic blocking keys selection in record linkage process [Benkhlaed *and al.* 2020]. This

approach is based on the Bald Eagle Search (BES) algorithm. This section explains with details the proposed solution and how we successfully adapted the BES algorithm to solve the blocking key selection problem.

In what follows, we describe the preliminaries and the notation used in this section. First, a dataset D is represented as follows: $D = \{t_1, t_2, \ldots, t_i\}$ with $(1 \leq i \leq |M|)$, where M is the size of the dataset (number of instances). Each $t_i \in D$ represents a tuple which is characterized by a set of attributes $A = \{Att_1, Att_2, \ldots, Att_j\}$ such that $(1 \leq j \leq |N|)$ and N is the number of attributes in the dataset.

F is a set of possible blocking functions where $F = \{f_1, f_2, \ldots, f_k\}$ with $(1 \leq k \leq |F|)$. Each $f_k \in F$ can be considered as a general blocking function that can be applied on any attribute or it can be a specific blocking function for a specific attribute. For instance, the "Exact-Value ()" function can be applied on any field (Att_j, attributes) from the dataset D, while the function "Extract-Zip ()" is specified for the address attributes. The result of applying a blocking function on an attribute for a specified tuple is the blocking key value (BKV). A BKV is denoted as $BK_{i,k}(t_i) = (Att_j, f_k)$.

In our work, the problem is to select the best possible subset of blocking keys ($BK_{i,k}$) among all the possible generated blocking keys using the blocking functions in F. The best blocking keys are the ones that bring the most possible true matches together into the same blocks.

4.3.1 Behavior of bald eagles when hunting

First, we explain the behavior of eagles during hunting which can be summarized in three main phases. The first step in bald eagles hunting strategy is to select a hunting space by tracking other birds to places with a high concentration of fish. Once selecting a hunting space, they begin to search for available fish (dead or alive) on that area. When an available prey is spotted, Bald Eagles swoop directly and snip it from the water.

4.3.2 Adapting BES algorithm to automatic blocking key selection

Bald Eagle Search optimization algorithm was recently proposed to solve the optimization problems. It was constructed using both swarm and evolutionary techniques [Alsattar *and al.* 2019].

As mentioned above, our aim in this section is to adapt the BES algorithm to the blocking key selection problem. This latter can indeed be modeled as a feature selection problem. The initial population is a group of subset of features, i.e., blocking keys. Thereby, each member of a population represents a competing subset of blocking keys. In addition, the subsets do not necessarily have the same sizes or elements. Populations are regenerated using the BES algorithm.

The fitness of each member of a population is calculated using the RL approach proposed in [Benkhaled *and al.* 2019] in a wrapper way. In this approach, K-Modes is used as an indexing step by clustering the data using only the blocking keys that are the currently selected subset of features. The best blocking keys in terms of fitness are the ones in which K-Modes groups the most duplicate records together when they are used as clustering attributes. As a result, the fitness function is the Pair Completeness (PC) parameter. PC measures the number of the detected duplicates by an RL approach using the selected blocking keys.

Our proposed approach can be summarized by the following points:

— Generate all possible blocking keys list.
— Choose the first population which is a subset of features from the previously generated blocking keys list.
— Run the BES algorithm for T iterations on the previously generated population in a wrapper method with the Pair completeness as the fitness function.
— The best member from the last population is selected as the best subset of features to be used as blocking keys.

4.3.3 Candidate keys generation

The candidate keys list is the one from which the BES algorithm's initial population will be selected. Before generating the candidate keys list, an essential preprocessing step cannot be overlooked. It is, in fact, a question of cleaning the set A. in other words, it is necessary to eliminate the attributes of bad quality from the set A. Two parameters are used to calculate the overall quality of an attribute. First, completeness represents the percentage of null value concerning specified attributes [Pipino *and al.* 2002]. We used the NBC (null-based completeness) measurement where completeness is measured using equation (4.4). Using this method, the value 1 represents the best result and 0 the worst. All the attributes that have a completeness value under predefined threshold are eliminated from the candidate blocking keys list generation aren't used during the blocking keys list generation process.

$$Completeness(Att_j) = 1 - \frac{\text{number of null values in } Att_j}{\text{Number of instances in D}} \qquad (4.4)$$

The second parameter is the cardinality of an attribute. Cardinality represents the number of distinct values for a specified attribute. In the RL process, the attributes with very low cardinality are not suitable to be used as blocking keys. For example, using the sex attribute as a blocking key divides the data into only 2 blocks (M/F). As a result, in our approach, the attributes with very low cardinality are eliminated from the candidate blocking keys list generation.

Once the attributes of bad quality get eliminated; for each dataset D, different blocking keys can be generated depending on the domain of the dataset and the type of attributes. We used a set of functions F to generate candidate keys like First4Chars (Attributes), Concatenation (), Soundex (Attribute), Last4Chars (Attribute), and NYSIIS (Attribute).

The following table shows some of the different functions used to generate the possible blocking keys list. Other specific functions were used for each dataset are not mentioned in the table. For example, "Extract-Number ()" is a function used to extract the restaurant number from the address field in the case of the restaurant dataset.

Table 4.1 – *The used functions for the blocking keys generation*

Function	Description
Soundex(Attribute), NYSIIS (Attribute)	Phonetic encoding algorithms [Holmes & McCabe 2002]).
First_N_Chars (Attribute)	Extract the first N characters from an attribute field.
Last_N_Chars (Attribute)	Extract the last N characters from an attribute field.
Numerical (Attribute)	Extract the numerical value from a string.
Remove-SP (Attribute)	Remove special characters from a string.
Exact-Value (Attribute)	Use the attribute value without modification

4.3.4 BES Pseudo Code

For the implementation of BES for the features selection problem, the population is a set of arrays, each one contains a number of integers, and each integer represents the index of one generated blocking key.

Algorithm 4.1 represents the pseudo-code of BES algorithm for features selection.

Algorithm 1. BES for feature selection.

```
1: Inputs: Dataset D, Number of iterations T, Number of
2:         solutions M, BES parameters: c1, c2,α, R.
3: Outputs: The best subset of blocking keys.
4: Function BES
5:    Initialize a population of M member.
6:    Initialize the best solution: Pbest = All features.
7:    While (i <= T) do
8:    For each (member P in pop)) do
9:              Pnew = Calculate (Pnew) using equation 4.5.
10:        IF (f (Pnew) > F (P)) do
11:             P = Pnew
12:        IF (f (Pnew) > F (Pbest)) do
13:             P = Pnew
14:        End IF
15:        End IF
16:        End For
17:    For each (member P in pop)) do
18:             Pnew = Calculate (Pnew) using equation 4.6.
19:        IF (f (Pnew) > F (P)) do
20:             P = Pnew
21:        IF (f (Pnew) > F (Pbest)) do
22:             P = Pnew
23:        End IF
24:        End IF
25:        End For
26:     For each (member P in pop)) do
27:             Pnew =Calculate (Pnew) using equation 4.7.
28:        IF (f (Pnew) > F (P)) do
29:             P = Pnew
30:        IF (f (Pnew) > F (Pbest)) do
31:             P = Pnew
32:        End IF
33:        End IF
34:        End For
35:   End While.
36:   Return Pbest.
37:  End Function.
```

The first step is to generate the initial population which is a selected subset of blocking keys from the previously generated list. Then, we start the space selecting phase (lines 8-16), where for each member of the population, we calculate a new object (P_{new}) from it using equation 4.5. α is a variable that takes a value between 1.5 and 2. R is a generated number with $R \in [0, 1]$ and P_i denotes the current position [Alsattar *and al.* 2019].

Once (P_{new}) is calculated, its new obtained features get cleaned, all the continues values get rounded into integers and all the integers that are outside the range of $[0, N]$ (N is the number of the generated blocking keys) get replaced by a another feature (blocking key index).

The next step is to measure the fitness using the features of P_{new} and compare it to the performance of both P and P_{best} if it's better than them. Then, they both get replaced by P_{new}.

In the searching space phase (lines 17-25), for each member of the population, a new object P_{new} gets calculated using equation 4.6. X and Y are two numbers to represent the spiral movement of the eagle towards the swooping area. Finally, the swooping (lines 26-34), it's where the eagle search for their final target in a spiral way. For each object in the population, P_{new} gets calculated using equation 4.7.

$$P_{new} = P_{best} + \alpha * r * (P_{mean} - P_i) \tag{4.5}$$

$$P_{new} = P_i + y(i) * (P_i - P_{i+1}) + x(i) * (P_i - P_{mean}) \tag{4.6}$$

$$P_{new} = rand * P_{best} + x(i) * (P_i - C_1 * P_{mean}) + y(i) * (P_i - C_2 * P_{best}) \tag{4.7}$$

All in all, the general bald eagle search optimization algorithm can be summarized in the following flow diagram (Figure 4.5)

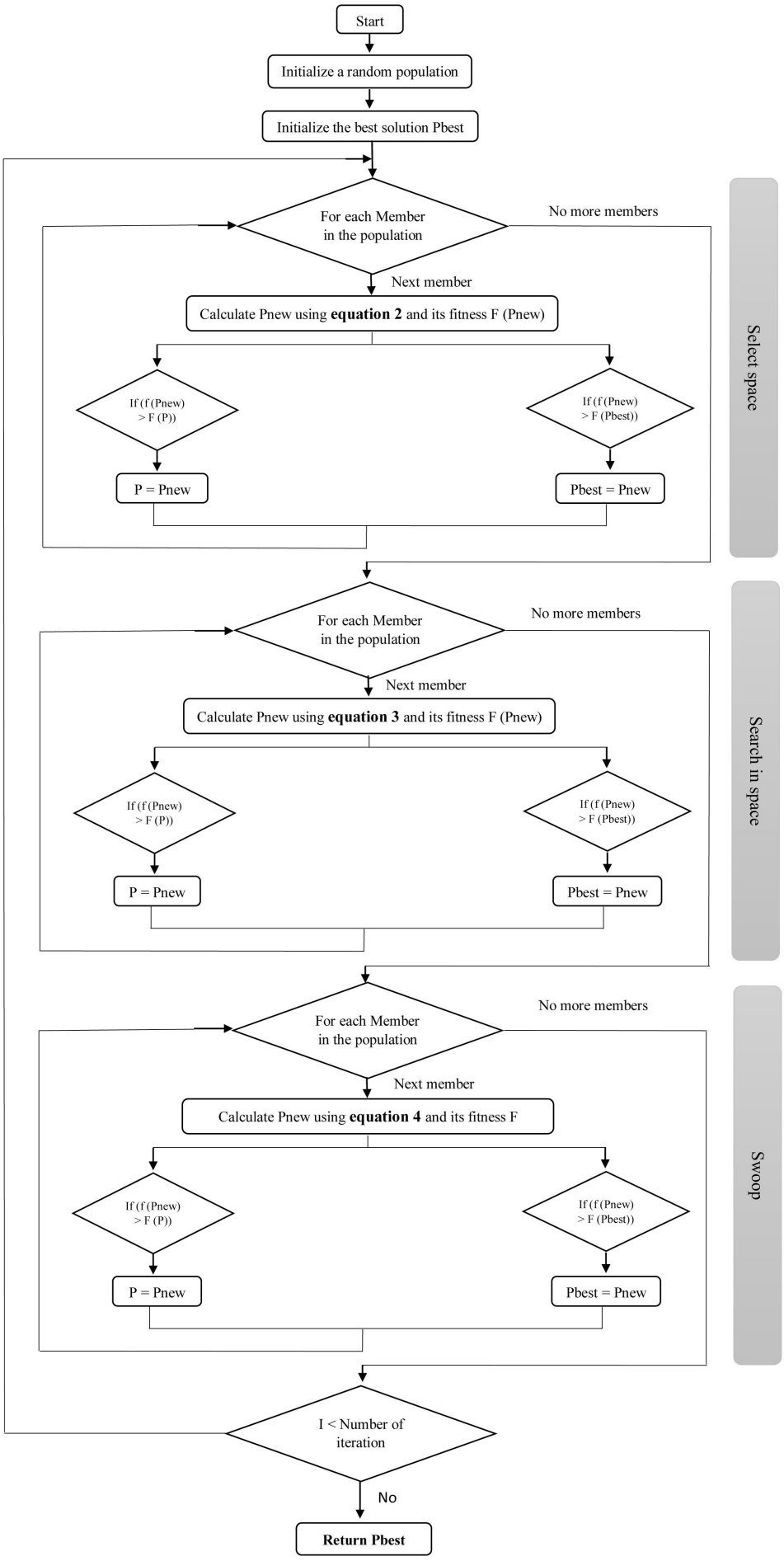

Figure 4.5 – *BES flow diagram*

4.3.5 K-Modes based Record Linkage

As previously mentioned, in order to test the performance of each subset from the selected features we used the K-Modes based Record Linkage approach proposed in the previous section [Benkhaled *and al.* 2019]. In this approach, the data get clustered into blocks using only the blocking keys as clustering attributes, which are in our case the selected features. Once the clustering is done, matching between the records in the same cluster is done using a string similarity metric like the Jaro-Winkler similarity. The number of the detected duplicated values is expressed using the Pair Completeness (PC) parameter (Equation 5.1). PC, in this case, is used as the fitness function in the BES algorithm.

4.4 Block sizes control

In this section, we propose a mechanism to control the size of the generated blocks by the K-Modes based Record Linkage [Benkhaled *and al.* 2020a]. K-Modes based RL divides the data into a number of blocks using only the blocking keys as clustering attributes instead of using all the dataset attributes. Doing that helps to improve the execution time and bringing more duplicates into the same cluster because BKVs contains the most useful information about the records. The advantage of the K-Mode based RL is the ability to create non-overlapping blocks where each entity gets assigned to one block only, which is not the case of other blocking techniques where the same entity can be assigned to different blocks like the case of Q-gram indexing or the suffix array-based indexing.

To control the size of the blocks, our approach uses as an input the K-Modes generated blocks. Then, all the blocks that are larger than a predefined MAX_{size} maximum size get divided into new blocks using the standard blocking with another blocking key. We keep dividing the generated blocks with a size larger than MAX_{size} until no block B with a

size $|B| > MAX_{size}$ is remaining.

Once the splitting phase is done, we start merging all the blocks that have a size less than a predefined value MIN_{size}. Each block b_I gets merged with the next smallest one b_j if they check the following conditions: (1) $|b_i| + |b_j| \leq MAX_{size}$ and (2) the similarity between the modes of bi and b_j is maximized. Compared to the approach of [Fisher *and al.* 2015], in which, they calculate the similarity of two blocks based on the similarity between their respective blocking keys. In our case, the similarity between the two blocks is represented by the similarity of their respective modes.

To calculate the similarity between the modes of two blocks, we use existed string similarity metrics like the Jaro-Winkler similarity where a complete similarity between all the attributes of the modes gets calculated.

4.4.1 Splitting blocks

Algorithm 4.2 represents the first step of our approach, which is blocks splitting. As an input, we have the predefined maximum block size MAX_{size} and the set of blocks generated by the K-Mode based RL (B^+) that has a size larger than the aforementioned fixed threshold.

The function Split starts by creating a new set of blocks B^{new} using the standard blocking with a new blocking key (line 6). Then, for each member of the newly created blocks, we see if its corresponded size is larger than MAX_{size} or smaller than $MINsize$. If it's larger than we insert it into B^+, otherwise, it gets inserted into the set of blocks that will be returned (B^{min}) (lines 7 to 11). All the aforementioned steps get repeated until no block remaining in B^+ (line 4).

4.4.2 Merging blocks

Algorithm 4.3 represents the second step in our block sizes controlling approach. Its purpose is to merge all the blocks that have a size under

Algorithm 4.2. Splitting blocks.

1: **Inputs:**
   - Maximum block size : MAX_{size}
   - Minimum block size : MIN_{size}
   - K-Modes blocks collection with a size $\geq MAX_{size}$: B^{+}
   - Blocking key List: K
2: **Outputs:** Blocks with size $\leq$ (MIN_{size}): (B^{min}).
3: **Function** Split
4: **While** $(B^{+} \neq \emptyset)$ **do**
5: **For each** (member B in (B^{+})) **do**
6: B^{new} = Standard-Blocking (B, new K)
7: **For each** (member C in (B^{new})) **do**
8: **IF** (C >= (MAX_{size})) **do**
9: B^{+}.Insert (C)
10: **Else IF** (C <= (MIN_{size})) **do** $B^{min}.Insert\,(C)$
11: **End If**
12: **End For**
13: **End For**
14: **End While**
15: **Return** B^{min}
16: **End Function.**

a specified threshold ($MINsize$). As an input we have: (1) the two parameters (MAX_{size}) and ($MINsize$). (2) The set of blocks generated by the K-Modes based RL with a size under $MINsize$ (B^{-}) and the block generated from the previous step which is block splitting (B^{min}). The first step is to merge both B^{min} and B^{-} into one sorted set named B^{merged}, where the blocks from B^{min} and B^{-} get sorted increasingly following their sizes (line 4). Next, for each block in B^{merged} we look for the block with the most similar mode and which achieves the sum condition (line 7). Once that is done, the current block gets merged with its closest block (line 8). If the new merged block B_{ij} is smaller than $MINsize$, then, it gets inserted into B^{merged} as a new block (line 10). Else, it gets inserted into the set that will be returned containing all the blocks with the correct size (line 12 and line 16).

The overall approach can be summarized in Algorithm 4.4, where we have the blocks generated by the K-Mode based RL as an input with the two parameters: MAX_{size} and $MINsize$. The first step is block preparation

Algorithm 4.3. Merging blocks.

Inputs:
- Maximum block size : MAX_{size}
- Minimum block size : MIN_{size}
- K-Modes blocks collection with a size $\leq MIN_{size}$: B^{-}
- Block generated after the splitting Phase : B^{min}
- K-Modes blocks collection with a correct size : $B^{K\_Correct}$
- Blocking key List : K

Outputs: Blocks with correct size: $B^{correct}$
Function merge
$B^{merged} = Sorted(B^{-} \, U \, B^{min})$ //sorted by size increasingly
While ($B^{merged} \neq \emptyset$) **do**
For each (member B in (B^{merged}) **do**
B_j = FindClosest (B, $B^{merged} \, U \, B^{K\_Correct}$)
/* find the block (B_j) with the most similar mode to B in $B^{merged} \, U \, B^{K\_Correct}$ where $|b_i| + |b_J| \leq MAX_{size}$ */
B_{ij} = MergeBlocks (B, B_j)
IF ($B_{ij} < (MIN_{size})$) **do**
B^{merged}.Insert (B_{ij})
Else
$B^{correct}$.Insert (B_{ij})
End For
End While
Return $B^{correct}$
End Function.

Algorithm 4.4. Block sizes control

Inputs:
- Maximum block size : MAX_{size}
- Minimum block size : MIN_{size}
- K-Modes blocks collection: BK.
- Blocking key List : K

Outputs: Blocks with correct size:
Begin
B^{-} , $B^{+}, B^{K\_Correct}$ = Prepare (BK, MIN_{size}, MAX_{size})
$B^{min} = Split\,(B^{-}, MAX_{size}, MIN_{size}, K)$
$B^{correct}$ = Merge $(B^{-}, B^{min}, MAX_{size}, MIN_{size}, K)$
Return ($B^{correct} \, U \, B^{k\_correct}$)
End.

(line 4), in which, all the blocks get assigned to three different sets (B^- : blocks with size under the fixed threshold, B^+ : blocks with size above the fixed threshold, $B^{k-correct}$: blocks with the correct size). Next, we apply the aforementioned functions for splitting and merging blocks (lines 5-6). Finally, the result is the union between the sets returned by the merge function and the block generated directly by K-Modes with the correct size.

4.5 K-Modes Based RL For Arabic data

The first step in our methodology is to add a pre-processing step to deal specifically with the Arabic language specifications. Like removing Titles from names as: (د.) which means the Title "Doctor" and (السيد) which refers to "Mr." In English in addition to other Arabic titles. Another important pre-processing step in our methodology is to remove the white spaces in the Arabic composite names like: (عبد الرحمن). Doing that helps to have a single translated name "Abdurrahman" which helps the phonetic encoding task.

4.5.1 Transliteration

As mentioned in the introduction, K-modes based RL uses only the blocking keys as clustering attributes. In order to use this approach on Arabic datasets, our proposed solution is to add an additional transliteration step before applying the phonetic encoding function on the selected attributes to create the blocking keys. Once the attributes are transliterated into their English representation, the phonetic encoding functions can be applied on the obtained results and generates the final blocking keys that the data will be divided into blocks based on them.

We chose to use the transliteration approach proposed in [Habash *and al.* 2007], the authors proposed an extension to the Buckwalter that was used in [Alian *and al.* 2017]. The suggested transliteration includes non-ASCII characters so the Arabic punctuation can be easily

remembered. The applied rules in our methodology are summarized in Figure 4.6.

Using the aforementioned rules transforms names like "لوهيبي" to "lwhyby" and "مصطفاوي" to "mSTfawy" which make it easy to apply any existing phonetic encoding function on the translated name such as Soundex in order to create an efficient blocking key.

Arabic	Transliteration	Arabic	Transliteration	Arabic	Transliteration
ء	'	ز	z	◌ً	.
آ	Ā	س	s	◌ٌ	a
أ	Â	ش	š	◌ٍ	~
ؤ	ŵ	ص	S	◌ّ	ũ
إ	Ă	ض	D	◌ُ	ã
ئ	ŷ	ط	Ď	◌ِ	ĩ
ا	A	ع	ς	◌ِ	i
ب	b	غ	γ	◌ُ	u
ة	h	ف	f		
ت	t	ق	q		
ث	θ	ك	k		
ج	j	ل	l		
ح	h	م	m		
خ	x	ن	n		
د	d	ه	h		
ذ	ð	ى	ý		
ر	r	ي	y		
و	w	ظ	t		

Figure 4.6 – *Punctuation of Arabic consonant*

All in all, our proposed methodology can be summarized in Figure 4.7. We can see that the proposed approach is composed of 4 principal steps:

Preprocessing: In which the mentioned operation in section 3.1 are applied over the dataset.

Transliteration of attributes: During the step, the attributes selected to form the blocking keys get transliterated using the rules defined in in Figure 4.6.

Blocking keys generation: In this step, the blocking keys get generated using the attributes translated in the previous step using the Soundex phonetic encoding, Then k-Modes is executed over the data using only the blocking keys as clustering attributes for a fixed number of iteration.

Matching: Once the K-Modes blocks are created, matching between

the records of the same cluster is done using the Jaro Winkler similarity metric

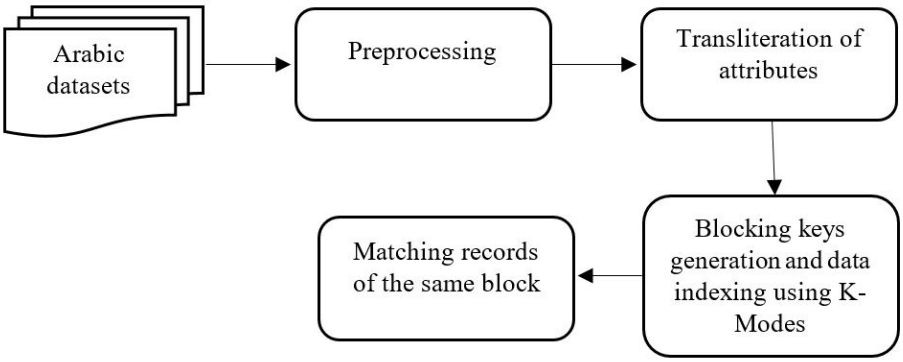

Figure 4.7 – *Proposed methodology (K-Modes based RL for Arabic datasets)*

4.6 Conclusion

In this chapter, we presented our 4 main contributions to the main challenges of Record Linkage. First, to reduce the search space and to avoid comparing each record in the dataset to all the others for matching. A new blocking technique was proposed, the idea was to use the K-Modes algorithm to cluster the data into a set of blocks using only the blocking keys as the clustering attributes. Doing that will bring all the similar records together into the same cluster where matching is done only between the records pairs of the same block. To reduce even more the number of comparisons we used a set of filtering techniques that removes more unlikable matches from the blocks before starting the comparison step. Moreover, a parallel implementation based on Map-Reduce was proposed for the K-Modes based RL. Using a parallel architecture made the K-Modes based RL more scalable in the case of a very large dataset.

The second contribution was about the selection of the blocking keys, we proposed an approach that automatizes the selection of attributes that are included in the creation of blocking keys. We used a recently proposed meta-heuristic (BES) after noticing the high performance of meta-heuristics in resolving the NP-Hard problem which is the case of the automatic blocking key selection.

The third contribution was made after noticing that most of the existed blocking techniques have to mechanism to control the size of the generated blocks. We proposed a solution that merges the small blocks and divides the larger ones into the desired sizes.

Finally, the last contribution was about dealing with non-Latin characters. We chose to add a transliteration step that transforms the Arabic characters into their correspondence ones in the English alphabet. This helped to apply K-Modes directly on Arabic datasets.

Results and discussion

Sommaire

5.1 Introduction

Each of the proposed approaches presented in the previous chapter is evaluated in this one. Section 5.2 presents the metrics used during this chapter to evaluate the performance of each approach. Section 5.3 shows the used benchmark during the evaluation of our contributions. In section 5.4, two types of experiments are presented in order to evaluate our K-Modes based RL. Section 5.5, shows the experiments that were done in order to test the efficiency of the proposed automatic blocking keys selection approach. Section 5.6 presents the experimental study for the proposed block sizes control approach. Finally, section 5.7 is dedicated to testing the adaptation of K-Modes based RL to the Arabic language.

5.2 Metrics

In the Record Linkage community, three main parameters are used to measure the performance of an RL process. These metrics are used in the evaluation of all the proposed solutions during this chapter.

Definition 1: Reduction ration (RR) (equation 4.1), this metric is used to measure how much the comparisons space is reduced by using the blocking technique.

$$RR = 1 - \frac{\text{Number of compared pairs}}{\text{all record pairs in D}} \tag{5.1}$$

Definition 2: Pair completeness (PC) (equation 4.2) measures the true positives, which is the percentage of the true matches detected by the proposed RL approach.

$$PC = \frac{\text{Number of detected record pairs}}{\text{Number of true matches in D}} \tag{5.2}$$

Definition 3: F-Score (equation 3) is used to measure the harmonic mean between the previous two parameters.

$$F_{score} = \frac{2 * PC * RR}{PC + RR} \tag{5.3}$$

5.3 DATASETS

In order to evaluate our proposed solutions. A set of real-world datasets was used. These datasets were selected since they were used to evaluate most of the existing RL approaches in the literature. The first one is the restaurant dataset. It contains records about restaurants in the United States collected from the Fodor and Zagat guides. The second one is the combination of the DBLP and ACM datasets which are both bibliographic citations. The combination of these two sets generates 4910 records including 2224 true matches. The third one is the Amazon/Google-Product dataset. It contains information about products collected from different sources (Amazon and Google). The fourth one is Cora. It contains publications in the machine learning area with duplicated records.

In addition to the aforementioned datasets, we used a fifth Arabic one that was used in [Ouhab *and al.* 2017]. This dataset contains duplicated information about students in our faculty (First name, Last name, and Address). These duplicated values were generated randomly using typographical errors and word rearrangement. Table 5.1 summarizes information about the used datasets in this chapter.

Table 5.1 – *Datasets used in the experiments*

Dataset Name	**Type**	**Records**	**True matches**
Restaurant	Deduplication	864	112
DBLP/ACM	Linkage	4910	2224
Amazon/GoogleProduct	Linkage	4598	1300
Cora Dataset	Deduplication	9763	17184
Arabic Dataset	Deduplication	3000	1000

5.4 K-Modes Based RL with Map-Reduce

To assess our proposed approach, we used one of the most famous implementations of the Map-Reduce paradigm which is Hadoop. We set up a 4 nodes Hadoop cluster with one NameNode and three DataNodes, each node has an Intel CPU, 4GB of memory and 64-Bit Ubuntu 18.04 LTS as an operating system that runs with JAVA and Hadoop 2.9.1 installed. In order to deal with our cluster limits, we changed the default configuration by allocating 1GB of RAM to each Map and Reduce execution. Table 5.2 summarizes our cluster configuration.

Table 5.2 – *Haddop Cluster Configurations*

Node	**CPU**	**RAM**	**Hadoop**	**Operating System**
NameNode	Intel i5-3450	4GB	2.9.1	Ubuntu 18.04
DataNode1	Intel i7-2600	4GB	2.9.1	Ubuntu 18.04
DataNode2	Intel i7-2600	4GB	2.9.1	Ubuntu 18.04
DataNode3	Intel i7-2600	4GB	2.9.1	Ubuntu 18.04

In our experiments we used the combination of the DBLP and ACM datasets, We used this dataset because it was used in most of the traditional record linkage works. Specifically, regarding pair completeness where a gold standard already exists for this dataset.

For testing the scalability of our RL approach, we increased the size of the dataset by duplicating its original records until we reached the needed size. The first step in our experiments was the generation of the blocking keys. We used two BKs, the first was the Soundex encoding of the publication's title concatenated with the publication's year and the second one was the NYSIIS encoding of the publication's title. We did not use the authors' names as part of the blocking keys after noticing that this attribute is an error-prone one where most of them are even wrong or incomplete. Two types of experiments were evaluated, the first one was to evaluate the quality of our proposed approach, and the second one was for its scalability. To compare our proposed K-Modes based RL with other

blocking techniques, we used the Restaurant dataset. Our choice of using this dataset in comparison was after noticing that this dataset is used to test most of the existing blocking techniques in the literature.

5.4.1 Quality experiments

Firstly, we evaluated the Reduction Ration of our proposed parallel blocking based on the K-Modes algorithm and the effectiveness of the Adaptive filtering as a post-processing step to blocking. We varied the number of clusters from 10 to 25 clusters and measured the obtained results concerning RR each time with and without filtering. The reduction ration with filtering is mentioned as Filtered Reduction Ration (FRR).

As we can see from the obtained results in Figure 5.1, the reduction ration without filtering keeps improving in a nearly linear way with the augmentation of the clusters' number. This is obvious because when dividing the data into more blocks (clusters) we reduce more the number of records comparisons. For example, with 10 clusters the obtained result was 0.77 concerning RR, this value is improved to 0.86 with 25 clusters.

When it comes to the Reduction Ration with filtering (FRR), the obtained results are stable even with varying the number of clusters which shows the high effectiveness of the filtering techniques by eliminating as much as possible the unlikable matches from the matching step.

For the pair completeness experiments, we varied the clusters' number from 5 to 25 and we measured the percentage of the detected true matches which is PC using three different string similarity metrics (Jaro-Winkler (JWS), Jaccard similarity (JS) and the Edit distance (ED)). The threshold of similarity was fixed to 0.9 for JWS and JS and 3 Edits for ED. For example, if the similarity between two records is above 0.9 then they are declared as true matches. The similarity between records was measured

using only three attributes which are: Blocking key 1, blocking key 2 and the publication's title. The obtained results are depicted in Figure 5.2.

As we can see, for the three different similarity metrics, the highest obtained result is with 10 clusters where 99% of the true matched records are detected. This value gets degraded with the augmentation of the cluster's number reaching 94% with the Jaro-Winkler similarity. This degradation concerning PC is explained by the fact that with more clusters we lose more true matches because some true matches can be assigned to different clusters and will not be compared to each other. In general, the obtained result shows that the Jaro-Winkler similarity is a suitable metric for this case.

From the discussed results above, we also notice that there is a trade-off between RR and PC. With more RR, we reduce more the number of comparisons which means less execution time. But at the same time, we lose Pair completeness because some true matches get assigned to different clusters. This problem can be solved by using the filtering techniques where the reduction ration results are stable even when the number of clusters is changed.

Figure 5.3 shows the stability of F-Score which is the trade-off between RR and PC because of using the filtering techniques.

The experiments results are also depicted in Table 5.3.

Table 5.3 – *Results for the DBLP/ACM datasets with the three different similarity metrics*

		ED (T= 3 edits)		**JS (T $\geq$ 0,9)**		**JWS (T $\geq$ 0,9)**	
K	**RR (FRR)**	**PC**	**F-S**	**PC**	**F-S**	**PC**	**F-S**
10	0.77 (0.98)	0.99	0.99	0.99	0.99	0.99	0.98
15	0.82 (0.99)	0.96	0.97	0.94	0.96	0.97	0.98
20	0.85 (0.99)	0.90	0.94	0.91	0.95	0.95	0.97
25	0.86 (0.99)	0.88	0.93	0.90	0.95	0.94	0.96

Figure 5.1 – *Reduction ration results with/without filtering (RR/FRR) for the DBLP/ACM datasets*

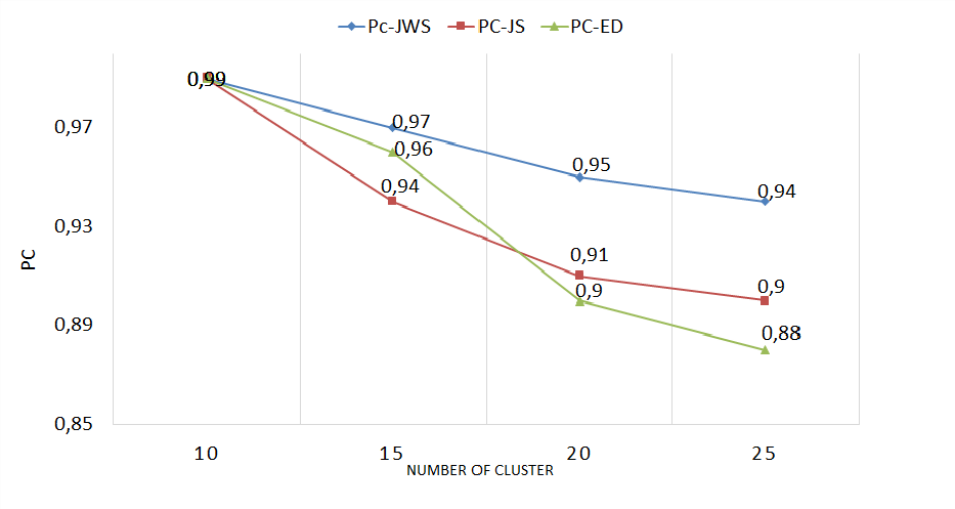

Figure 5.2 – *Pair Completeness results for the DBLP/ACM datasets*

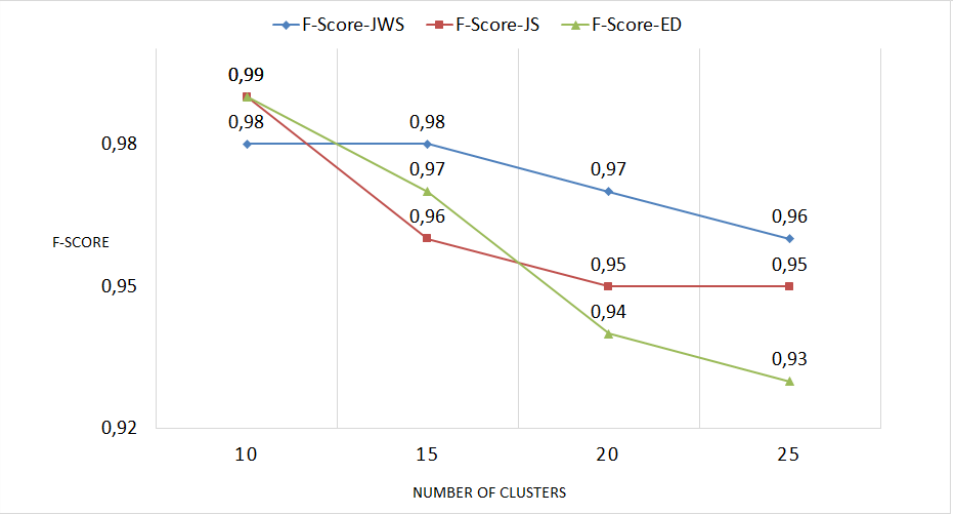

Figure 5.3 – *F-Score results for the DBLP/ACM datasets*

In order to test the difference concerning the execution time when using all the dataset attributes and the case of using only the blocking keys with the publication's title, we measured the execution time of matching in both cases.

The results are depicted in Figure 5.4, where with more clusters we have less execution time because dividing the data into more blocks reduces the number of comparisons. As expected, matching between the records using all the dataset attributes takes a longer time than using only the blocking keys as matching attributes. We believe that in the case of a very large dataset this difference would be much clearer.

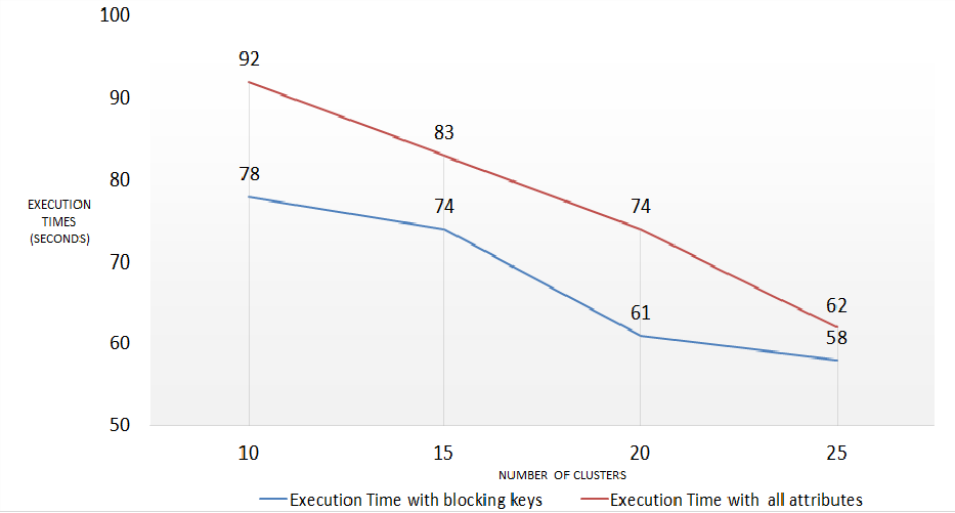

Figure 5.4 – *Execution time evolution*

Compared to other blocking techniques' results published in several surveys and papers for the famous restaurant dataset like in [Christen 2011] [Baxter *and al.* 2003], our proposed approach stands as a powerful one in terms of reduction ration where it averages 0.99 (with filtering) which is better than most of the existing indexing techniques. Regarding Pair Completeness, an average of 0.92 was achieved which is competitive with the other approaches (Figure 5.5.

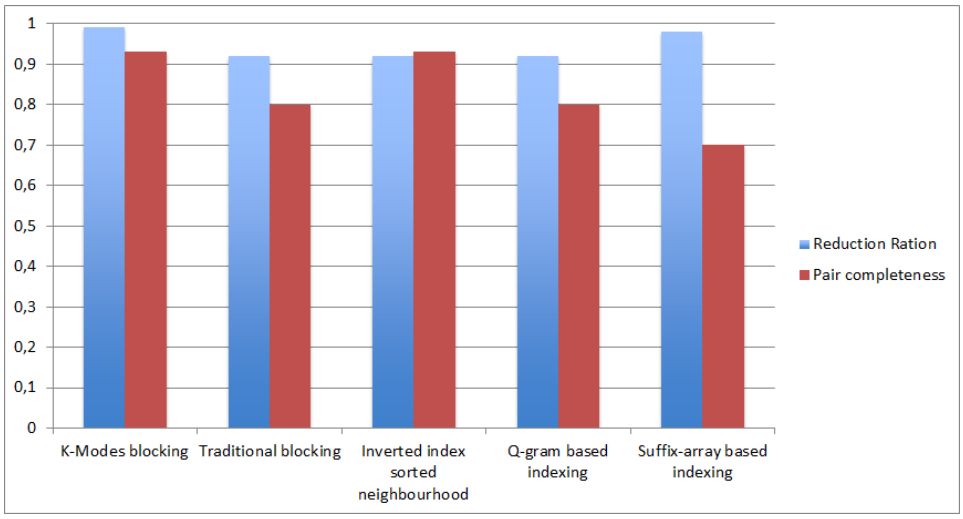

Figure 5.5 – *Comparative chart for the restaurant dataset*

5.4.2 Scalability experiments

The second type of experiments was to test the scalability of our proposed record linkage approach, we doubled the DBLP/ACM dataset size each time in order to test the performance in the case of a large dataset. We generated four different datasets (DBLP/ACM $x500$, DBLP/ACM $x1000$, DBLP/ACM $x1500$, DBLP/ACM $x2000$) in which DBLP/ACM $x2000$ reached over 10 millions records. After that, we examined the execution time of all the processes including blocking key generation, clustering, and matching. The obtained results are depicted in Figure 5.6 where we can see the execution time evolves in a nearly linear way with the data size. We believe that with an appropriate Hadoop cluster, the execution time will be much better since we used only 3 data nodes in our experiments due to material limitations.

From the experiments, we also noticed that sometimes a few clusters can contain a large number of records compared to the others (Figure 5.7), which can generate some latency problems when large blocks are directed to the same reducer while small blocks to others. This is generally due to the bad choice of the initial modes. This problem is known as the data skew problem where an unfair distribution is done among the reducers. In Figure 5.7, we can see that most of the instances were assigned to cluster

3 which will be directed to the same reducer. However, in the future, we plan to resolve this problem.

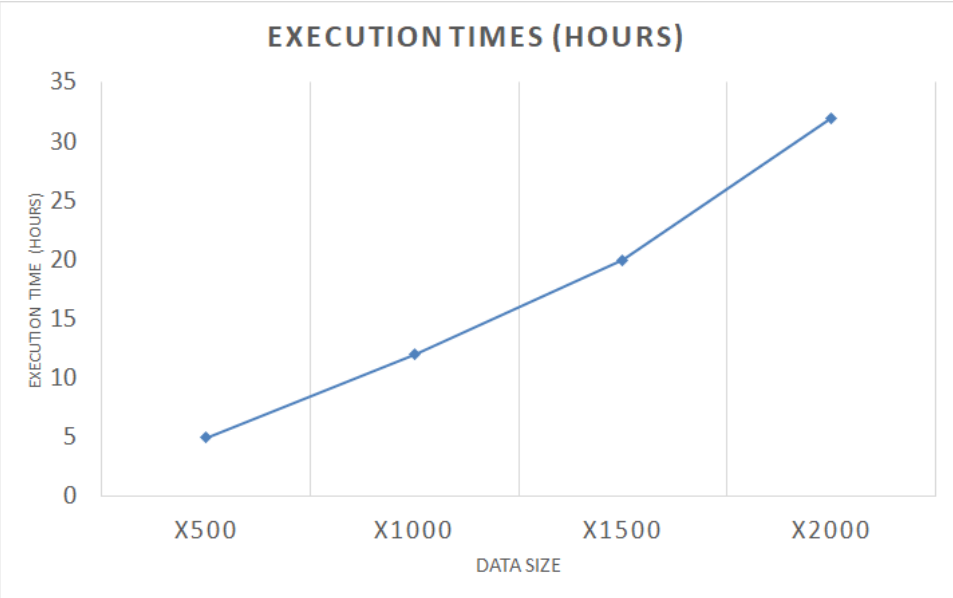

Figure 5.6 – *Execution time evolution with large datasets*

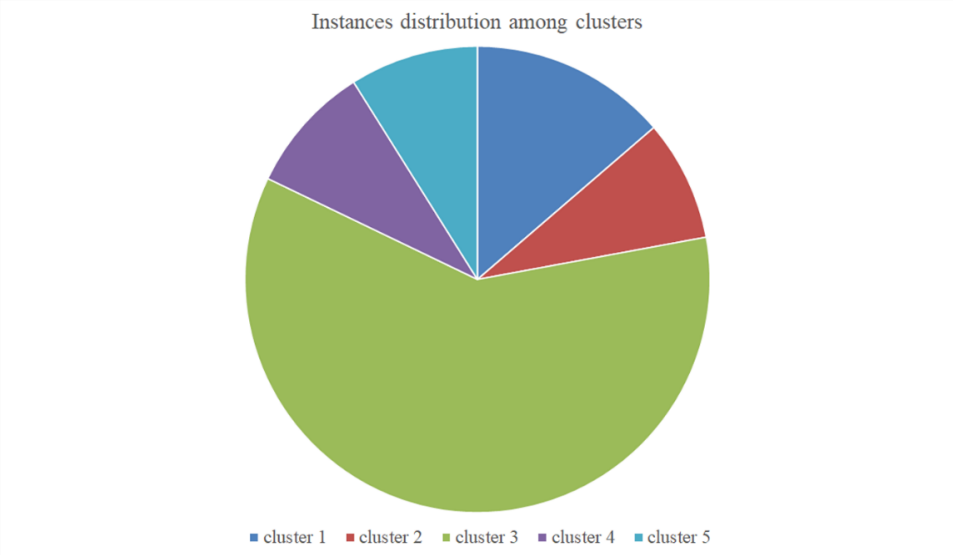

Figure 5.7 – *Data skew problem example*

5.5 Automatic blocking keys selection

In order to test the performance of our proposed automatic blocking keys approach, we used three famous real-world datasets "Restaurant", "The combination of the DBLP and ACM" and "Amazon/Google-Product" datasets. These datasets are usually used to evaluate the ef-

fectiveness of the automatic blocking keys selection in the RL community ([Shao & Wang 2018], [Kejriwal & Miranker 2013], [Christen 2011], [Köpcke *and al.* 2010a], [Kopcke *and al.* 2010b]). As a reminder from the previous section, restaurant dataset contains records about restaurants collected from the Fodor and Zagat guides. DBLP and ACM datasets are both bibliographic citations, the combination of these two sets generates 4910 records including 2224 true matches. Amazon/Google-Product dataset contains information about products collected from different sources (Amazon and Google). In addition to the aforementioned datasets, we used also the Arabic dataset.

Table 5.4 – *Experiments results*

Dataset	**measures**	**All attributes**	**Baseline**	**BES (T=4)**	**BES (T=8)**	**BES (T=16)**
Restaurant	RR	0.99	0.99	0.99	0.99	0.99
	PC	0.90	0.95	0.95	0.97	0.98
	F-Measure	0.94	0.94	0.94	0.97	0.98
DBLP/ACM	RR	0.99	0.99	0.99	0.99	0.99
	PC	0.83	0.93	0.97	0.98	0.99
	F-Measure	0.90	0.95	0.98	0.98	0.99
Amazon/Google-Product	RR	0.99	-	0.99	0.99	0.99
	PC	0.91	-	0.92	0.93	0.93
	F-Measure	0.95	-	0.95	0.96	0.96
Arabic dataset	RR	0.99	-	0.99	0.99	0.99
	PC	0.97	-	0.96	0.97	0.97
	F-Measure	0.98	-	0.97	0.98	0.98

The first round of experiments was done on restaurant dataset, it contains five attributes about each restaurant (Name, Addressee, City, Phone, and Type). 60 possible blocking keys were generated, including the Soundex encoding of the restaurant's name, NYSIIS encoding of the restaurant's name, the concatenation of the first four characters from the restaurant's name and the zip code from the addressee field and 57 other possible blocking keys using the functions discussed in the previous sections.

Once the generation of the possible blocking keys is done, we prepared the first random population for the BES algorithm. For that, we generated a set of random subsets of features (blocking key) of random size included between 0 and 60 which is the number of the generated blocking keys. When the maximum number of iterations is reached, the result of the execution is the best subset of blocking keys that gives the best PC. BES returned after 16 iterations the following keys (Table 5.5) as the best subset of blocking keys with best-obtained Pair completeness:

Table 5.5 – *BES best selected blocking keys for the restaurant dataset*

Iterations	**Selected blocking keys**
4	{[First4Chars(City)]+[First4Chars(Phone)], [Soundex(City)+Phone number], [Name]}
8	{[Phone Number], [First4Chars (Name) + First4Chars (ZipCode)], [First4(Name)+First4(Type)], [Soundex (City)]}
16	{[Phone Number], [First4Chars (Name) + First4Chars (ZipCode)], [Soundex (Name)], [First4Chars (Phone Number) + First4Chars (City)]}

The symbol "+" means a concatenation between two strings.

For the restaurant dataset, in order to compare our results with the other automatic blocking key selection techniques, we used the approach proposed in [Kejriwal & Miranker 2013] as a baseline approach.

From the obtained results in Table 5.4, we note that using the selected BKs by the BES algorithm (after 16 iterations) overcomes the results obtained in [Kejriwal & Miranker 2013], in which, they used the fisher discrimination to select the best blocking keys. At the same time, using the BES selected BKs as clustering attributes for the K-Modes based RL have better results than using all the attributes of datasets as BKs. An improvement of 4% concerning PC was gained after using the BES selected BKs.

In a second step, we tried to test the convergence of the BES algorithm following the number of iterations in the case of the restaurant dataset.

For that purpose, we varied each time the number of iteration from 4,8,12 to 16 iterations and we measured the evolution of the Pair completeness. Figure 4.13 shows the results of this experiment, in which we can see that the performance keeps improving until reaching 12 iterations. After that, the performances start to stabilize since the best results are already obtained.

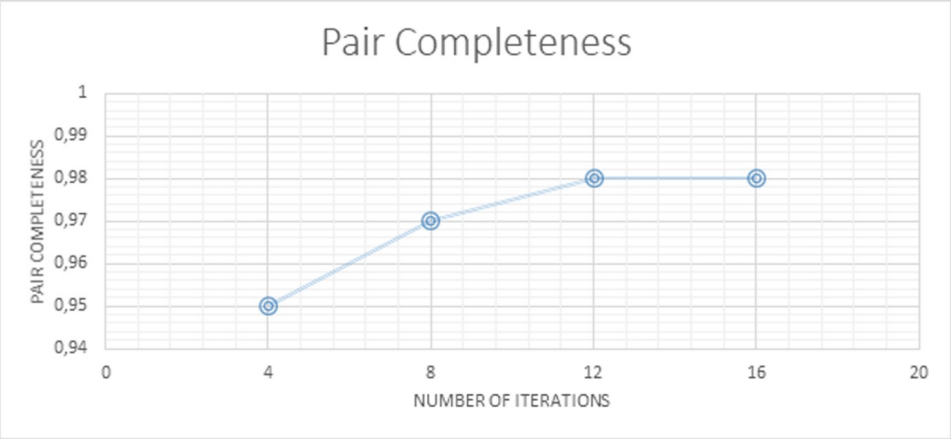

Figure 5.8 – *The convergence of BES by varying the iterations number with the restaurant dataset*

In a third phase, in order to show the importance of choosing the right blocking keys, we made a comparison between the results of the original K-Modes based RL where the blocking keys were selected manually and the result of K-Modes based RL with the blocking keys selected by BES. For that, we tried four different executions for the RL process. In the first execution (EX1), we used the blocking keys that were selected manually in section 4.1. The second execution (EX2) was done using the blocking keys selected by BES. For the third, fourth and fifth executions (EX3 & EX4 & EX5), we used some of the dataset attributes as BKs. After each execution, we measured the obtained Pair Completeness (number of detected true matches). Table 5.6 summarizes the used BKs for each execution.

From the obtained results in Figure 4.14, we can see the importance of choosing the right blocking keys on the performance of the RL process. Using the BKs selected by BES largely overcomes the other executions (EX3, EX4, and EX5) where random BKs where used. Compared to BKs

Table 5.6 – *Different executions and the associated blocking keys*

Execution	Used Blocking keys
Execution 1 (EX1)	{[Soundex (Name) + (Phone Number)], [NYSIIS (Name) + Extract-Number (Address)] }
Execution 2 (EX2)	{[Phone Number], [First4Chars (Name) + First4Chars (ZipCode)], [Soundex (Name)], [First4Chars (Phone Number) + First4Chars (City)]} (BES selected BKs)
Execution 3 (EX3)	{[Name]}
Execution 4 (EX4)	{[(Name)] + [(Phone)]}
Execution 5 (EX5)	{ First4Chars (Name) + [(City) + (Phone)]}

used in (EX2), we note an improvement of 2% regarding PC. This difference can be much clearer in the case of real-time RL, in which, the intervention of an expert to choose the right BKs is not possible due to time limitation. As a result, we can conclude that an automatic blocking key selection process is of a high importance in order to get satisfying results.

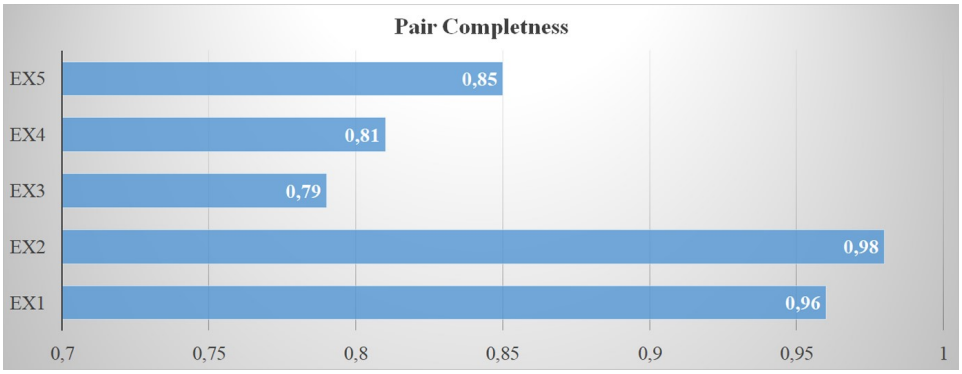

Figure 5.9 – *Pair completeness result with different execution parameters*

The second round of experiments was done on the DBLP/ACM dataset. This dataset contains information about bibliographic citations composed of 4 attributes (Title, Authors, Venue, and Year). 20 possible blocking keys were generated including the First author's name concatenated with the publication' year, the Soundex encoding of the title concatenated with the publication's years in addition to 18 more different blocking keys.

The initial population for the BES algorithm was composed of 20 random subsets of randomly selected features from the previously gener-

ated 20 blocking keys. The proposed approach in [Shao & Wang 2018] was used as a baseline approach for comparison in the case of the DBLP/ACM dataset.

From the obtained results (Table 4.5), we can see that the performance of the RL process using the blocking key selection by the BES algorithm largely overcomes using all the datasets attributes and the blocking keys selected by the baseline approach. The best-obtained result was 0.99 regarding completeness where 99% of the duplicate records were detected with an improvement of 6% compared to the baseline approach and 16% compared to using all the dataset attributes. Regarding the convergence in the case of the DBLP/ACM dataset, same as the restaurant dataset, the BES algorithm reaches its best results after 12 iterations. BES returned after 16 iterations the following keys:

— [Soundex (Title) + Year], [First4Chars (Author) + Year)], [First4Chars (Title) +First4Chars (Authors)], [Year].

The DBLP/ACM is considered as a large dataset compared to the first one (Restaurant), for that purpose, we measured the execution time evolution with augmentation of the iterations number. The obtained results are depicted in Figure 5.10, where we can see the execution time evolves in a nearly linear way with the number of iterations.

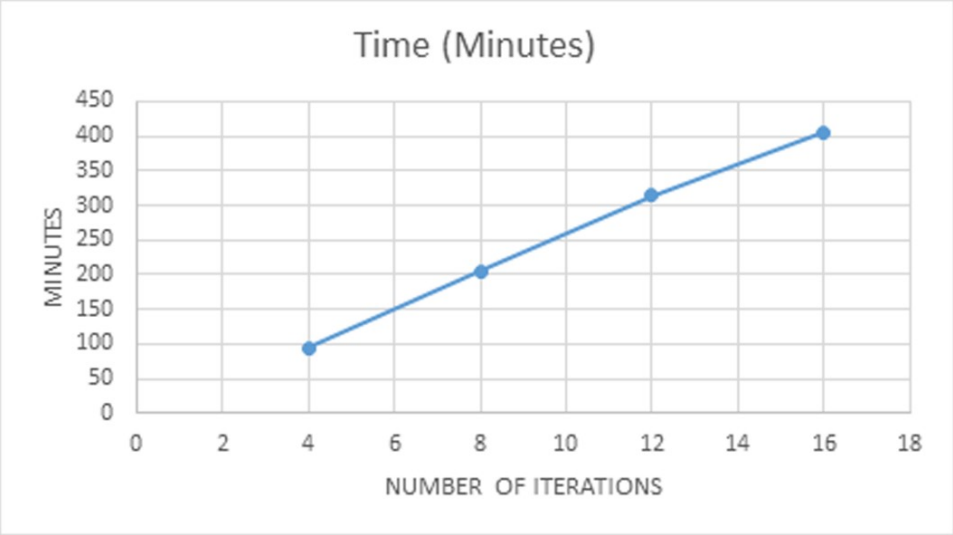

Figure 5.10 – *Time evolution by varying the iterations' number*

The third round of experiments was on the Amazon/Google-Product

dataset. This dataset includes four attributes (Product Name, Description, Manufacturer, and Price). 20 different blocking keys were generated using the functions mentioned in the previous section, Like: First4Chars (title) concatenated with the Price, First4Chars (Title) concatenated with the manufacturer and 18 other blocking keys. The BES initial population was initialized by 20 random subsets of features from the previously 20 generated blocking keys.

From Table 4.5, we can notice that using the BES selected features slightly improves the obtained results. After 16 iterations, PC improved from 0.91 using all the attributes as blocking keys to 0.92 using the selected blocking keys by BES. The best selected subset of blocking keys for the Amazon/GoogleProduct dataset returned by the BES algorithm was:

— [First4Chars (Title) + price)], [First4Chars (manufacturer) +First4Chars (price)].

For the Arabic dataset, 20 different blocking keys were generated . For example we used First4Chars (First name) concatenated with the home number from the address field, Soundex (First name), NYSIIS (Last name), Exact (First name) and 16 other blocking keys. The initial population was composed from 8 random subset of features from the 20 generated blocking keys.

From Table 4.5, after reaching 16 iterations, we can see that using the selected blocking keys by the BES algorithm gives the same results as using all the attributes of dataset as keys, where PC reaches 0.97. This dataset contains only 3 attributes which justify the fact that there is not a real improvement in the results.

Figure 5.11 shows the different values of F-Measure for the four tested datasets, we can see that using the BES selected blocking keys improves largely the RL's correspondent F-measure in the restaurant and DBLP/ACM dataset. On the other hand, we have a slight improvement

for the Amazon/GoogleProduct dataset and an equality for the Arabic dataset.

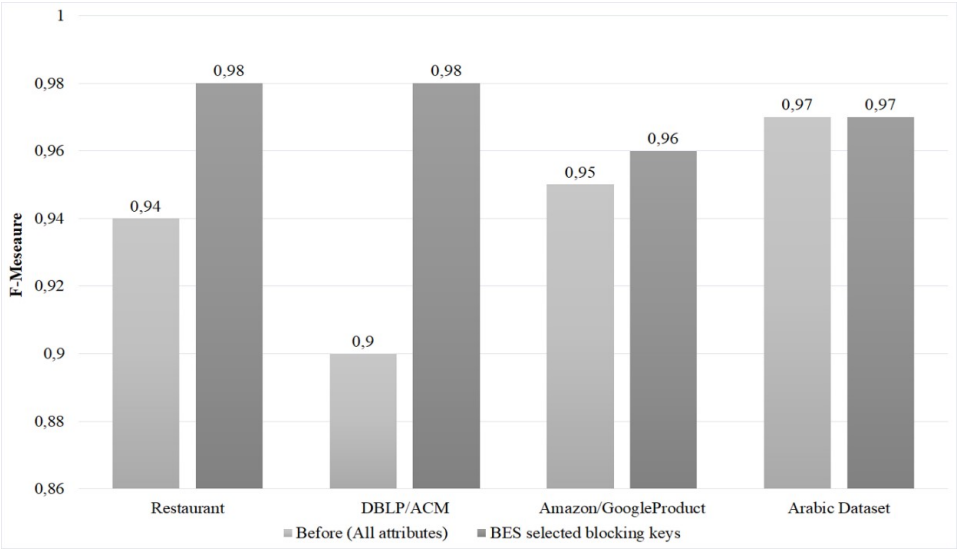

Figure 5.11 – *F-Measure values for the tested datasets Before/after blocking keys selection*

The last round of experiments was to analyze the number of the selected blocking keys by BES at each iteration. 5.12 shows the evolution of the selected blocking keys number. We can see that the progression does not have a linear evolution because the BES algorithm generates at each iteration new subsets randomly and good blocking keys can be ignored at one iteration and included in the next one.

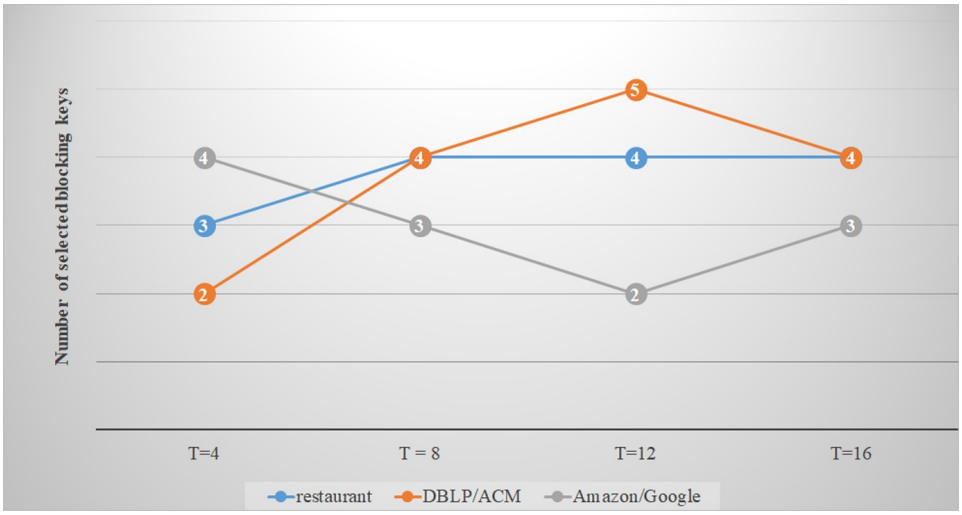

Figure 5.12 – *Evolution of the selected blocking keys number*

5.6 Block sizes control

This section describes the experimental study that has been done to evaluate our proposed approach on different real-world datasets and discusses the obtained results. To evaluate our proposition, 4 real-world datasets were used. Restaurant, DBLP/ACM, and the Arabic dataset.In addition to these three datasets, the Cora dataset is used.

The approach was implemented using JAVA and all our experiments were done on a machine with an Intel i5-9400F with 6 cores 2.90 GHz and 16 Gigabytes of memory.

In order to compare the performance of our blocks' size controlling against the K-Modes based RL without blocks' size controlling, we used the same settings that were used in section 4.2, in which, two blocking keys were used for the restaurant dataset. BK1: Soundex encoding of the restaurant's name concatenated with the phone number. BK2: NYSIIS encoding of the restaurant's name concatenated with the restaurant's number from the address field. For the DBLP/ACM dataset, two blocking keys were generated to use as clustering attributes (BK1: the publication's title encoded using Soundex concatenated with the publication' s year.

BK2: NYSIIS encoding of the publication's title).

We also provided a list of additional possible blocking keys for each dataset, these blocking keys can be used to divide the data in the case of clusters with a size larger than the predefined maximum size MAX_{size} as explained in section 3.1. For matching between the record pairs we used the Jaro-Winkler similarity with a threshold of 0.9. The filtering technique was not used in order to test the effect of controlling the block sizes on the Reduction Ration metric. Firstly, for the restaurant dataset. We varied the value of MAX_{size} and $MINsize$ (50-100, 100-200 and 100-250) and we measured RR, PC and F-Measure (FM) in all the cases. The obtained results are shown in the following table:

Table 5.7 – *Results for the restaurant dataset when controling block sizes*

	Min -Max								
	50-100			**100-200**			**100-250**		
K	**RR**	**PC**	**FM**	**RR**	**PC**	**FM**	**RR**	**PC**	**FM**
5	0.94	**0.87**	**0.90**	0.89	**0.89**	**0.89**	0.87	**0.91**	0.89
8	**0.94**	0.86	**0.90**	**0.91**	0.86	**0.89**	**0.89**	0.90	0.89
10	0.93	0.86	**0.90**	**0.91**	0.87	**0.89**	**0.89**	0.90	0.89

From the obtained results, we can see that choosing the value of the MAX_{size} parameter has a direct influence on the Reduction Ration. With small MAX_{size} values, we obtained a high performance concerning RR since the data get divided into smaller blocks which mean, a less number of comparisons between the records. The best RR result was obtained with $MINsize$ = 50 and MAX_{size} = 150 where RR reached 0.94. The value of RR keeps reducing with the augmentation of MAX_{size}.

On the other hand, choosing the value of $MINsize$ has a direct influence on Pair Completeness. When the $MINsize$ value is high, the clusters with a small size get merged which can bring more duplicates into the same cluster. In that case, more true positives will be detected during the matching phase. The best result concerning PC was obtained with

$MINsize$ = 100 and MAX_{size} = 250 where PC reached 0.91, which means that 91% of the duplicates records were detected.

To analyze the changes in the number of blocks (clusters) before and after block sizes controlling. We noted the number of the generated blocks after each execution. The obtained results are depicted in Figure 5.13. As we can see, when $MINsize$ and MAX_{size} parameters were fixed to 50 and 150 respectively, the number of blocks gets augmented. After that, the number of clusters keeps dropping down with the augmentation of $MINsize$ and MAX_{size}.

The first augmentation in the number of blocks was caused by the splitting of small blocks that were over MAX_{size} which was fixed only to 150. On the other hand, the later reduction in the number of clusters is explained by merging the blocks that have a size under $MINsize$.

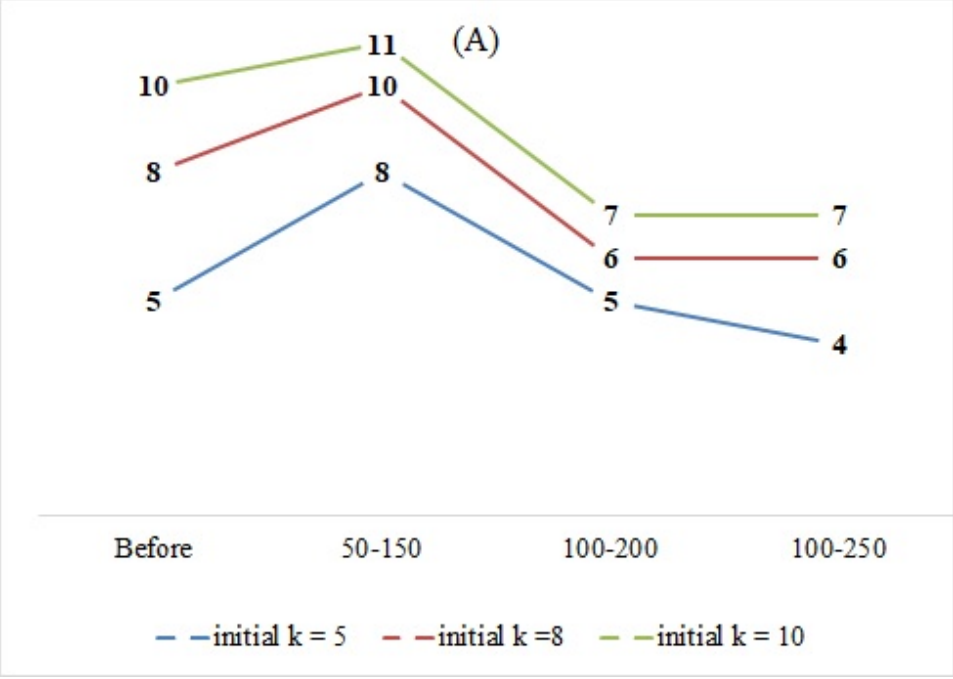

Figure 5.13 – *Evolution of the blocks' number and the execution time*

Concerning the DBLP/ACM dataset, we varied the value of MAX_{size} and $MINsize$ (100-400, 200-400 and 200-500) and we measured RR, PC,

and F-Measure in all the cases. Table 5.8 shows the obtained results.

Table 5.8 – *Results for the DBLP/ACM dataset*

	Min -Max								
	100-400			**200-400**			**200-500**		
	RR	**PC**	**FM**	**RR**	**PC**	**FM**	**RR**	**PC**	**FM**
10	0.97	**0.80**	0.87	0.97	0.86	0.91	0.97	0.87	*0.92*
13	0.98	**0.80**	0.88	**0.98**	**0.91**	**0.94**	**0.98**	**0.90**	***0.94***
15	**0.99**	**0.80**	**0.89**	**0.98**	0.90	0.93	**0.98**	**0.90**	***0.94***

Same as the restaurant dataset, the increasing in the value of $MINsize$ improves the results concerning PC where small blocks get merged, while choosing a small value for MAX_{size} increases the performance regarding the Reduction Ration.

Compared to K-Modes based RL without block sizes controlling, the obtained results concerning F-Measure are almost the same. Figure 5.14 represents a comparative chart of F-Measure for the obtained results when using the K-Modes based RL with/without block sizes controlling (case of $MINsize$ = 50 and MAX_{size} = 150 for the restaurant dataset and $MINsize$ = 250 and MAX_{size} = 500 for the DBLP/ACM dataset).

The third dataset (Arabic dataset) was used to test the scalability of the proposed block sizes controlling approach, Three different datasets were generated by replicating the original dataset records (D1: original x5, D2: original x10 and D3: original x15). We observed the execution time for both cases: K-Modes based RL without block sizes control and with block sizes control. The results are shown in Figure 5.15 . As can be seen, K-Modes based RL with block sizes control has a better performance concerning the execution time than K-Modes based RL without block sizes control. The gained time was during the matching phase since matching was done over blocks with controlled size which means fewer comparisons between the records. These obtained results also show the high efficiency of the

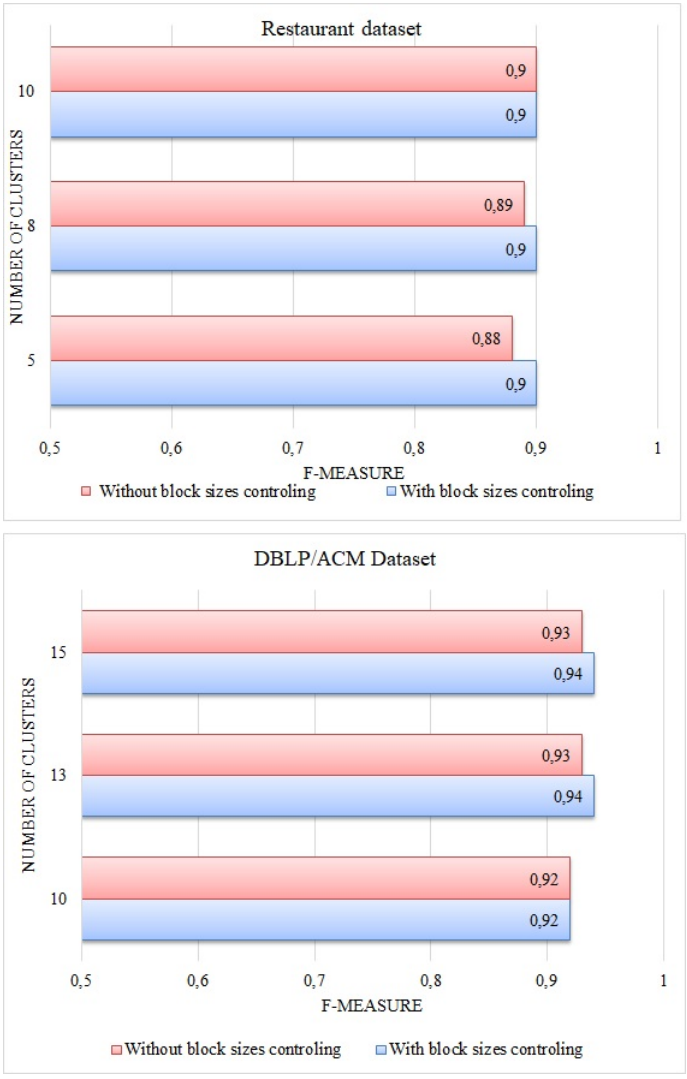

Figure 5.14 – *F-Measure comparison With/Without block sizes controlling*

proposed approach in making the K-Modes based RL more suitable for applications like real-time RL and privacy-preserving RL.

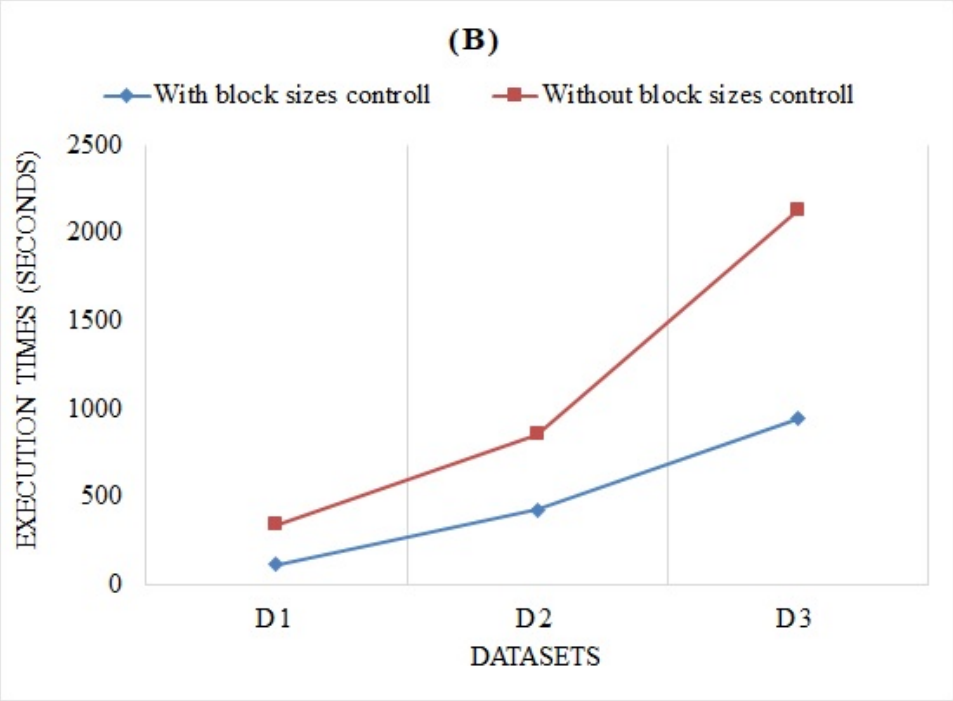

Figure 5.15 – *Execution time comparison*

5.7 K-Modes Based RL For Arabic datasets

In order to test our methodology, we used the real-world Arabic dataset that was used in section 4.3. As mentioned previously, this dataset is composed of three principal attributes: First Name, Last Name, and the Address attribute. It contains 3000 records including 1000 true matches. The duplicated records were generated randomly using typographical error and word rearrangement.

All our experiments were done on a machine with Intel i5-9400F with 6 cores 2.90 GHz and a memory of 16 Gigabytes and the methodology was implemented using JAVA.

To evaluate the effectiveness of our methodology on the K-Modes based Record Linkage, we tried two types of executions. In the first one, we did not use the transliteration process and we executed the K-Modes

directly on the three attributes of the dataset. Once the blocks are created matching was done between the records of the same block using the Jar-Winkler similarity with a fixed threshold of 0.9. Similarity calculations were done using all the three attributes of the dataset. In the second one, we used the transliteration process. K-Modes was executed on the data using only the blocking keys as clustering attributes. We generated two blocking keys. The first one is the Soundex encoding of the transliterated First Name concatenated with the Address number from the Address field. The second one is Soundex encoding of the transliterated Last Name. The obtained results for both executions are depicted in table 5.9.

Table 5.9 – *Experiments results*

		Without transliteration		**With transliteration**	
K	**RR**	**PC**	**F-Score**	**PC**	**F-Score**
5	0.99	0.936	0.966	0.97	0.98
10	0.99	0.898	0.945	0.937	0.937
15	0.99	0.866	0.927	0.953	0.967

From the obtained results 5.9, for both types of experiments (with/without transliteration), we noticed the same behavior mentioned in section 4.2 where PC reaches its highest result when the number of clusters is low. Then, PC keeps dropping down with the augmentation in the number of clusters. When the number of clusters is high, some true matches can get assigned to different clusters which explain the degradation in the PC results. Concerning RR, the results are consistence since the K-Modes based record Linkage uses a set of filtering technique in order to eliminate as much as possible record pairs from the matching step.

From the the comparison concerning PC in figure 5.16, we can see the transliteration step definitely improves the number of the detected pairs. Using the transliteration in creating the blocking keys improves the quality of the created blocks during clustering (bringing more true matches

together) which explains the improvement in the PC results compared to the first type of execution.

As expected, the execution time of the record Linkage approach increased after the addition of the transliteration step. From Figure 5.17, we can see that the execution time was doubled in the case of 10 and 15 clusters. The obtained results confirm that adding the additional step to adapt K-Modes based RL to the RL approach performs better in the case of a dataset of a small and medium-size. We believe that the execution time difference can be much clearer in the case of a very large dataset.

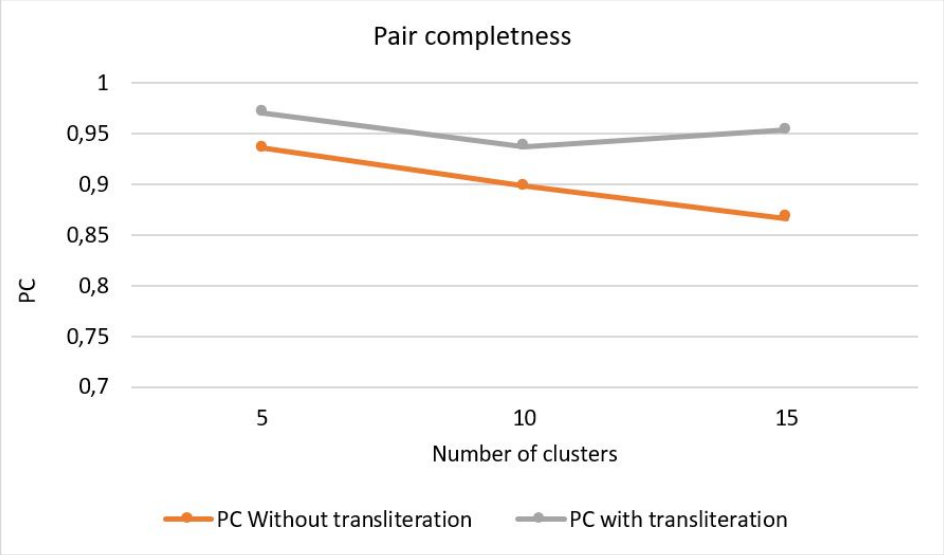

Figure 5.16 – *Pair completeness comparison*

5.8 Conclusion

In this chapter, we presented the experimental study that has been done to evaluate each of our contributions. First, we defined the metrics used in order to assess our works. Then we presented the datasets that were used for the evaluation of our contributions. The experiment results regarding the K-Modes based RL show the effectiveness of our proposition where it outperformed most of the blocking techniques of the literature. The experiments on a large dataset show the effectiveness of using a parallel implementation with Map-Reduce where the scalability of the ap-

Figure 5.17 – *Execution time comparison before/after transliteration*

proach was improved. The obtained results also showed the effectiveness of using the Bald Eagle Search algorithm in the selection of blocking keys even with a low number of iterations.

The proposed block sizes control approach also proved its effectiveness during the experiments, all the generated blocks sizes were into the desired sizes without losing any true matches. Finally, the proposed methodology to adapt the K-Modes based RL for the Arabic language showed promising results where most of the true matches records in the dataset were detected.

6

General conclusion

6.1 Conclusion

In the Big Data era, data is generated at an explosive rate every day. Big Data techniques and technologies like Hadoop helped organizations around the world to take decisive decisions such as huge investments in advertisements. However, making decisions based on data collected from everywhere without looking after its quality can have a negative impact on several aspects (Financial, business, reputation). According to the Gartner group, 40% of business initiatives didn't achieve their targeted benefits because of poor data quality.

Data quality problems can appears in different ways : missing values, duplicate values, referential integrity issues and many other ways. In this thesis, we focused on the duplicate values issue. Record Linkage is the process of detecting the tuples that represent the same real-world entity when integrating data coming from different sources.

In chapter 1, we presented the major challenges that have to be dealt with in order to face the duplicate values issue. The first challenge is to reduce the comparison space, comparing each record in the data to all the others can become computationally impossible in the case of large datasets. To do so, a blocking technique has to be used. Blocking divides the data into a set of blocks in a way that only the records clustered into the same block are compared to each other. The second major challenge is choosing the right blocking keys to divide the data. Traditionally, this task was done by a domain expert. However, the intervention of a domain expert can become a huge impediment in the face of real-time analyzing which is an important aspect of Big Data. As a result, proposing an automatic blocking keys selection approach is a pervasive challenge in the RL community. The third challenge is to control the sizes of the generated blocks, most of the blocking techniques in the literature were proposed without any mechanism to control the maximum and the minimum block sizes. Finally, dealing with non-Latin characters languages such as the Arabic language is also a pervasive challenge that has to be taken into consideration.

Chapter 2 was dedicated to present the major preliminaries and technologies that have to be known in order to better understand the tackled subject in this thesis. We defined Data warehouses and Big data and the impact of poor data quality on these two decision support systems. Then we presented the RL process and major Big Data challenges that have to be taken into consideration.

The current state of the art for Record Linkage was presented in chapter 3. Nearly 30 approaches were investigated in the chapter. A comparative study was also provided based on the following points: (1) Type of indexation. (2) Arabic language support. (3) Block sizes control. (4) Big Data support. (5) Automatic blocking keys selection which means if the proposed approach needs the intervention of an expert to select the blocking keys manually or the task is automatized.

Based on the challenges mentioned in Chapter 1 and the limitations of the approaches discussed in Chapter 3. In chapter 4, we presented a set of contributions for resolving all the challenges of RL. First, we presented a novel blocking technique with a parallel Map-Reduce implementation in order to deal with the new Big Data challenges. The proposed approach was based on the K-Modes algorithm as an indexing step and a filtering approach to reduce even more the comparison space. The obtained results showed the high efficiency and scalability of the proposed solution compared to the stat of art approaches.

Additionally, a novel approach was proposed in order to select automatically the blocking keys. The proposed approach was based on the Bald Eagle search meta-heuristic. We chose to use meta-heuristics because of their high performance when dealing with the NP-Hard problem which is the case of Blocking keys selection. The obtained were also satisfying where the best possible blocking keys were returned by BES. We also proposed a mechanism to control the size of the blocks generated by The K-Modes based RL. The idea was to split the large blocks and merge the small one until all the block respect the desired sizes. Finally, a solution was proposed in order to adapt our proposed K-Modes based RL with the Arabic languages.

Perspectives

As future work, we are interested in the following points :

— Propose a solution to the data skew problem discussed in section 4.2.

— The implementation of the proposed automatic blocking keys selection approach in a distrusted environment which will allow us to test with a higher number of iterations.

— The non-random initialization of the first population for the BES algorithm.

— In this thesis, we considered that the blocking functions in F used to generate the blocking keys are selected by a domain expert (Section 4.3). As future work, we may tackle the problem of the automatic blocking functions selection based on the domain of the dataset.

Bibliography

[Aizawa & Oyama 2005] 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, 2005.

[Alian *and al.* 2017] Marwah Alian, Ghazi Al-Naymat and Banda Ramadan. *Using transliteration with entity resolution for Arabic datasets*. In 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), pages 593–597. IEEE, 2017.

[Alsattar *and al.* 2019] HA Alsattar, AA Zaidan and BB Zaidan. *Novel meta-heuristic bald eagle search optimisation algorithm*. Artificial Intelligence Review, pages 1–28, 2019.

[Aqeel *and al.* 2006] Syed Uzair Aqeel, Steve Beitzel, Eric Jensen, David Grossman and Ophir Frieder. *On the development of name search techniques for Arabic*. Journal of the American Society for Information Science and Technology, vol. 57, no. 6, pages 728–739, 2006.

[Aranganayagi & Thangavel 2009] Sonia Aranganayagi and Kobi Thangavel. *Improved k-modes for categorical clustering using weighted dissimilarity measure*. International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 3, no. 3, pages 729–735, 2009.

[Bala *and al.* 1995] Jerzy Bala, Jeffrey Huang, Haleh Vafaie, Kenneth DeJong and Harry Wechsler. *Hybrid learning using genetic algorithms and decision trees for pattern classification*. In IJCAI (1), pages 719–724, 1995.

[Batini & Scannapieco 2016] Carlo Batini and Monica Scannapieco. *Data*

and information quality. Springer International Publishing., page 43, 2016.

[Baxter *and al.* 2003] Rohan Baxter, Christen Peter and Churches Tim. *A Comparison of Fast Blocking Methods for Record Linkage*. In Proceedings of the Workshop on Data Cleaning, Record Linkage and Object Consolidation at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

[Bello-Orgaz *and al.* 2016] Gema Bello-Orgaz, Jason J Jung and David Camacho. *Social big data: Recent achievements and new challenges*. Information Fusion, vol. 28, pages 45–59, 2016.

[Benkhaled & Berrabah 2019] Hamid Naceur Benkhaled and Djamel Berrabah. *Data Quality Management For Data Warehouse Systems: State Of The Art.* In the national Study Days on Computer Sciences Research, Saida, Algeria, 2019.

[Benkhaled *and al.* 2019] Hamid Naceur Benkhaled, Djamel Berrabah and Faouzi Boufares. *A novel approach to improve the Record Linkage process*. In 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), pages 1504–1509. IEEE, 2019.

[Benkhaled *and al.* 2020a] Hamid Naceur Benkhaled, Djamel Berrabah and Faouzi Boufares. *Block Sizes Control For an Efficient Real Time Record Linkage*. In 2020 The 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications, page To appear. IEEE, 2020.

[Benkhaled *and al.* 2020b] Hamid Naceur Benkhaled, Djamel Berrabah and Faouzi Boufares. *Data Warehouses and Big Data: How to Cope With Data Quality*. International Journal of Organizational and Collective Intelligence (IJOCI), vol. 10, no. 3, pages 1–13, 2020.

[Benkhlaed *and al.* 2020] Hamid Naceur Benkhlaed, Djamal Berrabah, Nassima Dif and Faouzi Boufares. *An Automatic Blocking Keys Selection For Efficient Record Linkage*. International Journal of Orga-

nizational and Collective Intelligence (IJOCI), vol. 11, no. 1, pages 53–70, 2020.

[Berkani *and al.* 2013] Nabila Berkani, Ladjel Bellatreche and Selma Khouri. *Towards a conceptualization of ETL and physical storage of semantic data warehouses as a service*. Cluster computing, vol. 16, no. 4, pages 915–931, 2013.

[Bilenko *and al.* 2006] 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.

[Bleiholder & Naumann 2009] Jens Bleiholder and Felix Naumann. *Data fusion*. ACM computing surveys (CSUR), vol. 41, no. 1, pages 1–41, 2009.

[Chandrashekar & Sahin 2014] Girish Chandrashekar and Ferat Sahin. *A survey on feature selection methods*. Computers & Electrical Engineering, vol. 40, no. 1, pages 16–28, 2014.

[Chaudhuri & Dayal 1997] Surajit Chaudhuri and Umeshwar Dayal. *An overview of data warehousing and OLAP technology*. ACM Sigmod record, vol. 26, no. 1, pages 65–74, 1997.

[Christen 2007] Peter Christen. *Towards parameter-free blocking for scalable record linkage*. ANU Research Publications, 2007.

[Christen 2011] Peter Christen. *A survey of indexing techniques for scalable record linkage and deduplication*. IEEE transactions on knowledge and data engineering, vol. 24, no. 9, pages 1537–1555, 2011.

[Cisco 2016] Visual Networking Index Cisco. *Global mobile data traffic forecast update, 2015–2020 white paper*. Document ID, vol. 958959758, no. 11, 2016.

[Clark 2004] David E Clark. *Practical introduction to record linkage for injury research*. Injury Prevention, vol. 10, no. 3, pages 186–191, 2004.

[Das Sarma *and al.* 2012] Anish Das Sarma, Ankur Jain, Ashwin Machanavajjhala and Philip Bohannon. *An automatic blocking mechanism for large-scale de-duplication tasks*. In Proceedings of the 21st ACM international conference on Information and knowledge management, pages 1055–1064, 2012.

[De Vries *and al.* 2009] Timothy De Vries, Hui Ke, Sanjay Chawla and Peter Christen. *Robust record linkage blocking using suffix arrays*. In Proceedings of the 18th ACM conference on Information and knowledge management, pages 305–314, 2009.

[Dean & Ghemawat 2008] Jeffrey Dean and Sanjay Ghemawat. *MapReduce: simplified data processing on large clusters*. Communications of the ACM, vol. 51, no. 1, pages 107–113, 2008.

[Dif & Elberrichi 2019] Nassima Dif and Zakaria Elberrichi. *An Enhanced Recursive Firefly Algorithm for Informative Gene Selection*. International Journal of Swarm Intelligence Research (IJSIR), vol. 10, no. 2, pages 21–33, 2019.

[Dif *and al.* 2018] Nassima Dif, Mohamed walid Attaoui and Zakaria Elberrichi. *Gene Selection for Microarray Data Classification Using Hybrid Meta-Heuristics*. In International Symposium on Modelling and Implementation of Complex Systems, pages 119–132. Springer, 2018.

[Dijcks 2012] Jean Pierre Dijcks. *Oracle: Big data for the enterprise*. Oracle white paper, page 16, 2012.

[Doan *and al.* 2012] AnHai Doan, Alon Halevy and Zachary Ives. Principles of data integration. Elsevier, 2012.

[El Akkaoui & Zimányi 2009] Zineb El Akkaoui and Esteban Zimányi. *Defining ETL worfklows using BPMN and BPEL*. In Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP, pages 41–48, 2009.

[El-Shishtawy 2013] Tarek El-Shishtawy. *A hybrid algorithm for matching arabic names*. arXiv preprint arXiv:1309.5657, 2013.

[Elziky *and al.* 2018] Mayada A Elziky, Dina M Ibrahim and Amany M Sarhan. *Improved Duplicate Record Detection Using ASCII Code Q-gram Indexing Technique*. Arabian Journal for Science and Engineering, vol. 43, no. 12, pages 7409–7420, 2018.

[Emary *and al.* 2015] Eid Emary, Hossam M Zawbaa, Kareem Kamal A Ghany, Aboul Ella Hassanien and Bazil Parv. *Firefly optimization algorithm for feature selection*. In Proceedings of the 7th Balkan Conference on Informatics Conference, pages 1–7, 2015.

[Fellegi & Sunter 1969] Ivan P Fellegi and Alan B Sunter. *A theory for record linkage*. Journal of the American Statistical Association, vol. 64, no. 328, pages 1183–1210, 1969.

[Fisher *and al.* 2015] Jeffrey Fisher, Peter Christen, Qing Wang and Erhard Rahm. *A clustering-based framework to control block sizes for entity resolution*. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 279–288, 2015.

[Fleuret 2004] François Fleuret. *Fast binary feature selection with conditional mutual information*. Journal of Machine learning research, vol. 5, no. Nov, pages 1531–1555, 2004.

[Gani *and al.* 2016] Abdullah Gani, Aisha Siddiqa, Shahaboddin Shamshirband and Fariza Hanum. *A survey on indexing techniques for big data: taxonomy and performance evaluation*. Knowledge and information systems, vol. 46, no. 2, pages 241–284, 2016.

[Geiger 2004] Jonathan G Geiger. *Data quality management, the most critical initiative you can implement*. Data Warehousing, Management and Quality, Paper, pages 098–29, 2004.

[Gravano *and al.* 2001] Luis Gravano, Panagiotis G Ipeirotis, Hosagrahar Visvesvaraya Jagadish, Nick Koudas, Shanmugauelayut Muthukrishnan, Divesh Srivastava*and al.* *Approximate string joins in a database (almost) for free*. In VLDB, volume 1, pages 491–500, 2001.

[Gu & Baxter 2004] Lifang Gu and Rohan Baxter. *Adaptive filtering for efficient record linkage*. In Proceedings of the 2004 SIAM International Conference on Data Mining, pages 477–481. SIAM, 2004.

[Guide 2013] Planning Guide. *Getting Started with Big Data*. Intel IT Center, 2013.

[Guyon & Elisseeff 2003] Isabelle Guyon and André Elisseeff. *An introduction to variable and feature selection*. Journal of machine learning research, vol. 3, no. Mar, pages 1157–1182, 2003.

[Habash *and al.* 2007] Nizar Habash, Abdelhadi Soudi and Timothy Buckwalter. *On arabic transliteration*. In Arabic computational morphology, pages 15–22. Springer, 2007.

[Halevy *and al.* 2005] Alon Y Halevy, Naveen Ashish, Dina Bitton, Michael Carey, Denise Draper, Jeff Pollock, Arnon Rosenthal and Vishal Sikka. *Enterprise information integration: successes, challenges and controversies*. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 778–787, 2005.

[Hernández & Stolfo 1995] Mauricio A Hernández and Salvatore J Stolfo. *The merge/purge problem for large databases*. ACM Sigmod Record, vol. 24, no. 2, pages 127–138, 1995.

[Hicham 2012] Gueddah Hicham. *Introduction of the weight edition errors in the Levenshtein distance*. arXiv preprint arXiv:1208.4503, 2012.

[Higazy *and al.* 2013] Azza Higazy, Tarek El Tobely, Ahmed H Yousef and Amany Sarhan. *Web-based Arabic/English duplicate record detection with nested blocking technique*. In 2013 8th International Conference on Computer Engineering & Systems (ICCES), pages 313–318. IEEE, 2013.

[Holmes & McCabe 2002] David Holmes and M Catherine McCabe. *Improving precision and recall for soundex retrieval*. In Proceedings. International Conference on Information Technology: Coding and Computing, pages 22–26. IEEE, 2002.

[Hsueh *and al.* 2014] Sue-Chen Hsueh, Ming-Yen Lin and Yi-Chun Chiu. *A load-balanced mapreduce algorithm for blocking-based entity-resolution with multiple keys*. In Proceedings of the Twelfth Australasian Symposium on Parallel and Distributed Computing-Volume 152, pages 3–9, 2014.

[Huang 1998] Zhexue Huang. *Extensions to the k-means algorithm for clustering large data sets with categorical values*. Data mining and knowledge discovery, vol. 2, no. 3, pages 283–304, 1998.

[Inmon 1992] WH Inmon. *Building the data warehouse, QED Information Sciences*. Inc., Wellesley, MA, 1992.

[Jensen *and al.* 2010] Christian S Jensen, Torben Bach Pedersen and Christian Thomsen. *Multidimensional databases and data warehousing*. Synthesis Lectures on Data Management, vol. 2, no. 1, pages 1–111, 2010.

[Jona & Nagaveni 2014] JB Jona and N Nagaveni. *Ant-cuckoo colony optimization for feature selection in digital mammogram*. Pakistan journal of biological sciences, vol. 17, no. 2, page 266, 2014.

[Kämpgen *and al.* 2012] Benedikt Kämpgen, Seán O'Riain and Andreas Harth. *Interacting with statistical linked data via OLAP operations*. In Extended Semantic Web Conference, pages 87–101. Springer, 2012.

[Karakasidis & Verykios 2012] Alexandros Karakasidis and Vassilios S Verykios. *Reference table based k-anonymous private blocking*. In Proceedings of the 27th Annual ACM Symposium on Applied Computing, pages 859–864, 2012.

[Karapiperis & Verykios 2015] Dimitrios Karapiperis and Vassilios S Verykios. *Load-balancing the distance computations in record linkage*. ACM Sigkdd Explorations Newsletter, vol. 17, no. 1, pages 1–7, 2015.

[Kejriwal & Miranker 2013] Mayank Kejriwal and Daniel P Miranker. *An unsupervised algorithm for learning blocking schemes*. In 2013 IEEE

13th International Conference on Data Mining, pages 340–349. IEEE, 2013.

[Khouri 2013] Selma Khouri. *Cycle de vie sémantique de conception de systèmes de stockage et manipulation de données*. PhD thesis, Ecole Nationale Supérieure de Mécanique et d'Aérotechique - Poitiers, France, 2013.

[Kolb & Rahm 2013] Lars Kolb and Erhard Rahm. *Parallel entity resolution with dedoop*. Datenbank-Spektrum, vol. 13, no. 1, pages 23–32, 2013.

[Kolb *and al.* 2012] Lars Kolb, Andreas Thor and Erhard Rahm. *Multi-pass sorted neighborhood blocking with MapReduce*. Computer Science-Research and Development, vol. 27, no. 1, pages 45–63, 2012.

[Köpcke *and al.* 2010a] Hanna Köpcke, Andreas Thor and Erhard Rahm. *Evaluation of entity resolution approaches on real-world match problems*. Proceedings of the VLDB Endowment, vol. 3, no. 1-2, pages 484–493, 2010.

[Kopcke *and al.* 2010b] Hanna Kopcke, Andreas Thor and Erhard Rahm. *Learning-based approaches for matching web data entities*. IEEE Internet Computing, vol. 14, no. 4, pages 23–31, 2010.

[Levenshtein 1966] Vladimir I Levenshtein. *Binary codes capable of correcting deletions, insertions, and reversals*. In Soviet physics doklady, volume 10, pages 707–710, 1966.

[Liu *and al.* 2012] Xiufeng Liu, Christian Thomsen and Torben Bach Pedersen. *Mapreduce-based dimensional etl made easy*. Proceedings of the VLDB Endowment, vol. 5, no. 12, pages 1882–1885, 2012.

[Maldonado *and al.* 2011] Sebastián Maldonado, Richard Weber and Jayanta Basak. *Simultaneous feature selection and classification using kernel-penalized support vector machines*. Information Sciences, vol. 181, no. 1, pages 115–128, 2011.

[McCaffrey 2009] James D McCaffrey. *Using the multi-attribute global inference of quality (MAGIQ) technique for software testing*. In 2009 Sixth

International Conference on Information Technology: New Generations, pages 738–742. IEEE, 2009.

[McCallum *and al.* 2000] Andrew McCallum, Kamal Nigam and Lyle H Ungar. *Efficient clustering of high-dimensional data sets with application to reference matching*. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 169–178, 2000.

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

[Nascimento *and al.* 2020] Dimas Cassimiro Nascimento, Carlos Eduardo Santos Pires and Demetrio Gomes Mestre. *Exploiting block co-occurrence to control block sizes for entity resolution*. Knowledge and Information Systems, vol. 62, no. 1, pages 359–400, 2020.

[Ouhab *and al.* 2017] Abdelkrim Ouhab, Mimoun Malki, Djamel Berrabah and Faouzi Boufares. *An unsupervised entity resolution framework for english and arabic datasets*. International Journal of Strategic Information Technology and Applications (IJSITA), vol. 8, no. 4, pages 16–29, 2017.

[Patil *and al.* 2011] PS Patil, Srikantha Rao and Suryakant B Patil. *Data integration problem of structural and semantic heterogeneity: data warehousing framework models for the optimization of the ETL processes*. In Proceedings of the International Conference & Workshop on Emerging Trends in Technology, pages 500–504, 2011.

[Pipino *and al.* 2002] Leo L Pipino, Yang W Lee and Richard Y Wang. *Data quality assessment*. Communications of the ACM, vol. 45, no. 4, pages 211–218, 2002.

[Rajkovic & Jankovic 2007] P Rajkovic and D Jankovic. *Adaptation and application of Daitch-Mokotoff Soundex algorithm on Serbian names*. In XVII Conference on Applied Mathematics, volume 12, 2007.

[Ramadan & Christen 2015] Banda Ramadan and Peter Christen. *Unsupervised blocking key selection for real-time entity resolution*. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 574–585. Springer, 2015.

[Redman 1998] Thomas C Redman. *The impact of poor data quality on the typical enterprise*. Communications of the ACM, vol. 41, no. 2, pages 79–82, 1998.

[Redman 2016] Thomas C Redman. *Bad data costs the US $3 trillion per year*. Harvard Business Review, vol. 22, pages 11–18, 2016.

[Redmond 2012] Wash Redmond. *The Big Bang: How the Big Data Explosion Is Changing the World*, 2012.

[Sagiroglu & Sinanc 2013] Seref Sagiroglu and Duygu Sinanc. *Big data: A review*. In 2013 international conference on collaboration technologies and systems (CTS), pages 42–47. IEEE, 2013.

[Salem *and al.* 2014] Aïcha Ben Salem*and al. Semantic Recognition of a Data Structure in Big-Data*. Journal of Computer and Communications, vol. 2, no. 09, page 93, 2014.

[Sarawagi & Bhamidipaty 2002] Sunita Sarawagi and Anuradha Bhamidipaty. *Interactive deduplication using active learning*. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 269–278, 2002.

[Sathi 2012] Arvind Sathi. Big data analytics: Disruptive technologies for changing the game. Mc Press, 2012.

[Shao & Wang 2018] Jingyu Shao and Qing Wang. *Active blocking scheme learning for entity resolution*. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pages 350–362. Springer, 2018.

[Skoutas & Simitsis 2007] Dimitrios Skoutas and Alkis Simitsis. *Ontology-based conceptual design of ETL processes for both structured and semi-structured data*. International Journal on Semantic Web and Information Systems (IJSWIS), vol. 3, no. 4, pages 1–24, 2007.

[Tao *and al.* 2015] Guo Tao, Ding Xiangwu and Li Yefeng. *Parallel K-modes algorithm based on MapReduce*. In 2015 Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC), pages 176–179. IEEE, 2015.

[Trujillo & Luján-Mora 2003] Juan Trujillo and Sergio Luján-Mora. *A UML based approach for modeling ETL processes in data warehouses*. In International Conference on Conceptual Modeling, pages 307–320. Springer, 2003.

[Vassiliadis *and al.* 2002] Panos Vassiliadis, Alkis Simitsis and Spiros Skiadopoulos. *Conceptual modeling for ETL processes*. In Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, pages 14–21, 2002.

[Vogel & Naumann 2012] Tobias Vogel and Felix Naumann. *Automatic blocking key selection for duplicate detection based on unigram combinations*. In Proceedings of the International Workshop on Quality in Databases (QDB), 2012.

[Yan *and al.* 2007] Su Yan, Dongwon Lee, Min-Yen Kan and Lee C Giles. *Adaptive sorted neighborhood methods for efficient record linkage*. In Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries, pages 185–194, 2007.

[Zaharia *and al.* 2016] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin*and al. Apache spark: a unified engine for big data processing*. Communications of the ACM, vol. 59, no. 11, pages 56–65, 2016.

[Zaidi *and al.* 2015] Houda Zaidi, Yann Pollet, Faouzi Boufarès and Naoufel Kraiem. *Semantic of data dependencies to improve the data quality*. In Model and Data Engineering, pages 53–61. Springer, 2015.

[Zaidi *and al.* 2016] Houda Zaidi, Faouzi Boufarès and Yann Pollet. *Net-*

toyage de données guidé par la sémantique inter-colonnes. In EGC, pages 549–550, 2016.

[Zhang *and al.* 2016] Lina Zhang, Liqiang Liu, Xin-She Yang and Yuntao Dai. *A novel hybrid firefly algorithm for global optimization*. PloS one, vol. 11, no. 9, 2016.

[Zikopoulos *and al.* 2013] P Zikopoulos, P De Ross, K Parasuraman, T Deutsch, J Giles and D Corrigan. *Harness the Big Data: The IBM Data Platform*, 2013.

Notations

ACO	Ant Colony optimization
BES	Bald Eagle Search
BK	Blocking Key
BKV	Blocking Key Value
BPEL	Business Process Execution Language
BPMN	Business Process Modeling Notations
CS	Cuckoo Search
DE	Deferential evolution
DQ	Data Quality
DSA	Data Staging Area
DSS	Decision support System
DW	Data Warehouse
ED	Edit Distance
ETL	Extract, Transform, and Load
FFA	Fire-Fly Algorithm
FRR	Filtered Reduction Ration
GA	Genetic Algorithms
HDFS	Hadoop distributed File System
IBM	International business machines
IBPK	Indexing Based on Parallel K-Modes
JS	Jaccard Similarity
JWS	Jaro Winkler Similarity
NYSIIS	New-York State Identification Intelligence System
OLAP	Online Analytical Processing
PC	Pair completeness

PMF	Parallel Matching with filtering
PeSOA	Penguin Search Optimization Algorithm
RL	Record Linkage
ROC	Rank Order Centroid
RR	Reduction Ratio
SCA	Sequential Covering Algorithm
SRP	Sorted Reduce Partitions
UML	Unified Modeling Language

Abstract

Organizations around the world lose trillions of dollars due to poor data quality problems. One of the main processes in the data quality field is the Record Linkage process. Record linkage is the process of identifying the tuples that refer to the same real-world entity. Without blocking, the RL process can end up by billions of comparisons when dealing with large datasets. Blocking reduces the number of comparisons by dividing the data into blocks in a way that only the records in the same block will be compared to each other.

In this thesis, we present the following contributions : (1) A new RL approach based on the K-Modes algorithm as a blocking step and a filtering technique as a post-processing step to blocking. The proposed RL approach is associated with a parallel implementation using Hadoop in order to face the Big Data challenges. (2) We propose an approach for the automatic blocking keys selection based on the Bald Eagle search algorithm. (3) We introduce a mechanism that controls the sizes of the K-Modes generated blocks. (4) We suggest a solution so that our K-Modes based RL can be used in the case of the Arabic language, the solution is to add an additional transliteration step to the RL process. The obtained results from experiments showed the efficiency of our propositions.

Keywords: Data quality, Record Linkage, Big Data, Map-Reduce, Data warehouses.

Résumé

Les Organisations du monde entier perdent des billions de dollars en raison de problèmes de qualité des données. L'un des principaux processus dans le domaine de la qualité des données est le processus de couplage d'enregistrements (Record Linkage). Le couplage d'enregistrements est le processus qui consiste à identifier les tuples qui se réfèrent à la même entité du monde réel. Sans blocage, le processus RL peut aboutir à des milliards de comparaisons lorsqu'il s'agit de grands ensembles de données. Le blocage réduit le nombre de comparaisons en divisant les données en blocs de manière à ce que seuls les enregistrements d'un même bloc soient comparés les uns aux autres.

Dans cette thèse, nous présentons les contributions suivantes : (1) Une nouvelle approche RL basée sur l'algorithme K-Modes comme étape de blocage et une technique de filtrage comme étape de post-traitement au blocage. L'approche RL proposée est associée à une implémentation parallèle utilisant Hadoop afin de faire face aux défis des Big Data. (2) Nous proposons une approche pour la sélection automatique des clés de blocage basée sur l'algorithme de recherche Bald Eagle. (3) Nous introduisons un mécanisme qui contrôle la taille des blocs générés par K-Modes. (4) Nous proposons une solution pour que notre RL basé sur K-Modes puisse être utilisé dans le cas de la langue arabe, la solution est d'ajouter une étape de translitération supplémentaire au processus RL. Les résultats obtenus par les expérimentations ont montré l'efficacité de nos propositions.

Mots-clés: Qualité des données, couplage d'enregistrements, Big Data, Map-Reduce, Entrepôts de données.

الملخص

تفقد المؤسسات حول العالم تريليونات الدولارات بسبب مشاكل جودة البيانات السيئة. في السنوات الأخيرة ، جعل الوعي بأهمية جودة البيانات أصحاب المصلحة يستثمرون الكثير من المال من أجل تحسين جودة البيانات المخزنة. واحدة من العمليات الرئيسية في مجال جودة البيانات هي عملية ربط السجلات. ربط السجلات هو عملية تحديد البيانات التي تشير إلى نفس الكيان الواقعي. بدون تجزئة، يمكن أن تنتهي عملية ربط البيانات بمليارات المقارنات عند التعامل مع مجموعة بيانات كبيرة. تؤدي تجزئة البيانات إلى تقليل عدد المقارنات عن طريق تقسيم البيانات إلى كتل بطريقة تتم فيها مقارنة السجلات الموجودة في نفس الكتلة فقط.

في هذه الأطروحة ، نقدم المساهمات التالية: (1) أسلوب جديد لربط البيانات يستند إلى خوارزمية كي-مودز كخطوة تجزئة وتقنية تصفية كخطوة بعد التجزئة. يرتبط النهج المقترح بالتنفيذ الموازي باستخدام تقنية ماب ريد يوس لمواجهة تحديات البيانات الضخمة. (2) نقترح نهجًا دو طابع تلقائي لاختيار مفاتيح التجزئة استنادًا إلى خوارزمية بحث النسر الأصلع. (3) نقدم آلية تتحكم في أحجام الكتل المولدة. (4) نقترح حلاً بحيث يمكن استخدام منهجنا المقترح في حالة اللغة العربية ، والحل هو إضافة خطوة تحويل صوتي إضافية إلى العملية. أظهرت النتائج التي تم الحصول عليها من التجارب كفاءة مقترحاتنا.

الكلمات المفتاحية : جودة البيانات, البيانات الكبيرة, ربط السجلات, مستودعات البيانات.