PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Milton Santos, s/n - Ondina
Salvador, BA, Brasil, 40170-110

https://pgcomp.ufba.br
pgcomp@ufba.br

Big Data is a trending topic that has gained attention in the business and academic environments. The term refers to the huge amount of data being generated every day in a variety of sources and formats. An expressive part of Big Data is in the format of text that can be used to solve various real life problems, such as spam detection, author identification, web pages classification and sentiment analysis. Text datasets are specially complicated since its high dimensionality can extend from vertical high dimensionality (high number of instances) to horizontal high dimensionality (high number of attributes). In order to extract useful knowledge from such high dimensional datasets, data analysis techniques must be able to cope with its new challenges: volume, velocity variety and variability. Fuzzy Rule-Based Classification Systems (FRBCS) have shown to effectively deal with the uncertainty, vagueness, and noise inherent to data. However, the performance of FRBCSs is highly affected by the increasing number of instances and attributes present in Big Data. Previously proposed approaches try to adapt FRBCSs to Big Data by distributing data processing with the MapReduce paradigm, by which the data is processed in two stages: Map and Reduce. In the Map stage, the data is divided into multiple blocks and distributed among processing nodes that process each block of data independently. In the Reduce stage, the results coming from every node in the Map stage are aggregated and a final result is returned. This methodology tackles vertical high dimensionality, but it does not approach datasets with simultaneous vertical and horizontal high dimensionality, as it is the case of text datasets. Horizontal high dimensionality reduction could be done by using common feature selection techniques, such as MI and Chi-squared. However, using such feature selection techniques may not be the best alternative since model accuracy might be affected by the loss of information when keeping only a subset of attributes. In this work, we deal with the aforementioned drawbacks by proposing Summarizer, an approach for building reduced feature spaces for horizontally high dimensional data. To this end, we carry out an empirical study that compares a well-known classifier proposed for vertical high dimensionality datasets with and without the horizontal dimensionality reduction process proposed by Summarizer. Our findings show that existing classifiers that tackles vertical Big Data problems can be improved by adding the Summarizer approach to the learning process, which suggests that an unified learning algorithm for datasets with a high number of instances as well as a high number of attributes might be possible.

MSC | 091 | 2020

On the use of fuzzy clustering to build fuzzy rule based systems to address big data

Pétala Gardênia da Silva
Estrela Tuy

UFBA

On the use of fuzzy clustering to build fuzzy rule based systems to address big data

Pétala Gardênia da Silva Estrela Tuy

Dissertação de Mestrado

Universidade Federal da Bahia

Programa de Pós-Graduação em
Ciência da Computação

Abril | 2020

Universidade Federal da Bahia

Instituto de Matemática

Programa de Pós-Graduação em Ciência da Computação

ON THE USE OF FUZZY CLUSTERING TO BUILD FUZZY RULE BASED SYSTEMS TO ADDRESS BIG DATA

Pétala Gardênia da Silva Estrela Tuy

DISSERTAÇÃO DE MESTRADO

Salvador

Abril de 2020

PÉTALA GARDÊNIA DA SILVA ESTRELA TUY

ON THE USE OF FUZZY CLUSTERING TO BUILD FUZZY RULE BASED SYSTEMS TO ADDRESS BIG DATA

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Tatiane Nogueira Rios

Salvador
Abril de 2020

Ficha catalográfica elaborada pela Biblioteca Universitária de Ciências e Tecnologias Prof. Omar Catunda, SIBI – UFBA.

T968 Tuy, Pétala Gardênia da Silva Estrela

On the use of fuzzy clustering to build fuzzy rule based systems to address Big Data / Pétala Gardênia da Silva Estrela Tuy. – Salvador, 2020.

182 f.

Orientadora: Prof.ª Dr.ª Tatiane Nogueira Rios

Dissertação (Mestrado) – Universidade Federal da Bahia. Instituto de Computação, 2020.

1. Fuzzy Clustering. 2. Big Data. 3. MapReduce. I. Rios, Tatiane Nogueira. II. Universidade Federal da Bahia. III. Título.

CDU 004.62

“On the use of fuzzy clustering to build fuzzy rule-based systems to address Big Data”

Pétala Gardênia da Silva Estrela Tuy

Dissertação apresentada ao Colegiado do Programa de Pós-Graduação em Ciência da Computação na Universidade Federal da Bahia, como requisito parcial para obtenção do Título de Mestre em Ciência da Computação.

Banca Examinadora

Tatiane Nogueira Rios

Prof.ª Dr.ª Tatiane Nogueira Rios (Orientadora-UFBA)

[illegible]

Prof. Dr. Matheus Giovanni Pires (UEFS)

[illegible]

Prof. Dr. Marcos Ennes Barreto (UFBA)

Dedico este trabalho a minha mãe, que sempre me incentivou a estudar e correr atrás dos meus sonhos.

ACKNOWLEDGEMENTS

I thank God first for giving me life and for giving me a chance to live with purpose.

I thank my mother and family for always supporting me and believing in me. I thank them for always understanding my divided attention when deadlines were close. I thank them for dreaming with me and for being always there when I needed. I thank my nephews for giving me moments of peace and distraction when I was exhausted.

I thank my life partner for believing in my work and for being proud of the smallest achievements. I thank him for trying to help me when I felt lost.

I am specially thankful for the good friends I found on the way (you know who you are). I am thankful for the ones that encouraged me to choose this challenging research area. If it was not for their encouragement and support, I would not feel strong and capable enough to take this leap. I thank the ones that dedicated their time and effort to help my statistician brain to get into this computer science world. I thank the ones that helped me see that I can be a ‘girl that codes’.

I thank my advisor, Tatiane Rios, for the excellent guidance during the whole time. I thank her for always making me grow and improve my academic skills. I thank her for always being understanding and patient.

I thank the Federal University of Bahia for making possible the developmentent of this work.

Finally, I thank all of those who helped me in this work, directly or indirectly.

A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê.

—ARTHUR SCHOPENHAUE

ABSTRACT

Big Data is a trending topic that has gained attention in the business and academic environments. The term refers to the huge amount of data being generated every day in a variety of sources and formats. An expressive part of Big Data is in the format of text that can be used to solve various real life problems, such as spam detection, author identification, web pages classification and sentiment analysis. Text datasets are specially complicated since its high dimensionality can extend from vertical to horizontal high dimensionality (high number of instances and attributes respectively). In order to extract useful knowledge from such high dimensional datasets, data analysis techniques must be able to cope with its new challenges: volume, velocity, variety and variability. Fuzzy Rule-Based Classification Systems (FRBCS) have shown to effectively deal with the uncertainty, vagueness, and noise inherent to data. However, the performance of FRBCSs is highly affected by the increasing number of instances and attributes present in Big Data. Previously proposed approaches try to adapt existing FRBCSs to deal with Big Data by distributing data processing with the MapReduce paradigm. This methodology tackles vertical high dimensionality, but it does not approach datasets with simultaneous vertical and horizontal high dimensionality, as it is the case of text datasets. Horizontal high dimensionality reduction could be done by using common feature selection techniques, such as MI and Chi-squared. However, using such feature selection techniques may not be the best alternative since model accuracy might be affected by the loss of information when keeping only a subset of attributes. In this work, we deal with the aforementioned drawbacks by proposing *Summarizer*, an approach for building reduced feature spaces for horizontally high dimensional data. To this end, we carry out an empirical study that compares a well-known classifier proposed for vertical high dimensionality datasets with and without the horizontal dimensionality reduction process proposed by *Summarizer*. Our findings show that existing classifiers that tackles vertical Big Data problems can be improved by adding the *Summarizer* approach to the learning process, which suggests that an unified learning algorithm for datasets with a high number of instances as well as a high number of attributes might be possible.

Keywords: Big Data, MapReduce, Fuzzy, FRBCS

CONTENTS

Chapter 1—Introduction 1

1.1 Motivation 3
1.2 Hypothesis and Objectives 4
1.3 Workflow 5

Chapter 2—Fuzzy Systems 7

2.1 Fuzzy sets 7
2.1.1 Membership functions 8
2.1.2 Operation on fuzzy sets 8
2.1.3 Fuzzy Relations 10
2.2 Computation With Fuzzy Rules 11
2.2.1 Linguistic Variables 12
2.2.2 Fuzzy Rules 13
2.2.3 Fuzzy Inference 13
2.3 Fuzzy Systems 14
2.3.1 Fuzzy Rule-Based Classification Systems 14
2.3.2 Fuzzy Inference System 15
2.3.3 Fuzzy Classification 16
2.4 Final Considerations 17

Chapter 3—Vertical Dimensionality Reduction 19

3.1 Big Data 19
3.2 MapReduce 21
3.3 *Apache Hadoop* 22
3.4 Adaptations of Traditional Machine Learning Algorithms for Big Data . 23
3.5 Fuzzy Rule Based Classification Systems for Big Data 26
3.5.1 CHI-BD 27

3.6 Final Considerations 29
3.7 Clustering Algorithms 30
3.7.1 Fuzzy C-Means 32
3.8 Principal Components Analysis (PCA) 35
3.8.1 Data Standardization 36
3.8.2 Covariance Matrix Computation 37
3.8.3 Eigenvectors and Eigenvalues Computation 37
3.8.4 Feature Vector Construction 38
3.9 Latent Semantic Analysis (LSA) 39
3.10 Final Considerations 39

Chapter 4—Summarizer 41

Chapter 5—Experiments 49

5.1 Experimental Framework 49
5.2 Number of Rules 50
5.3 Classification Performance 53

Chapter 6—Conclusions 57

6.1 Contributions 59
6.2 Limitations and Further Research 59

Chapter 7—Appendix 67

LIST OF FIGURES

2.1 Triangular membership function for a fuzzy set. 9
2.2 Membership function for the linguistic variable “Temperature”. 12
2.3 Representation of a Knowledge Base (KB) of a Fuzzy Rule-Based Classification System (FRBCS). 15

3.1 MapReduce work split between computing nodes. 22
3.2 Representation of a Hadoop MapReduce processing Model. 23
3.3 Percentage of explained variance by each Principal Component. 38

4.1 Workflow of *Summarizer* 47

5.1 Number of rules generated by the CHI-BD algorithm. 51

LIST OF TABLES

3.1 Simple texts example 33
3.2 Words count representation 33
3.3 Transposed matrix with TF-IDF representation. 34
3.4 Membership degrees generated by FCM. 34

4.1 Data matrix of Instances and Attributes ($I \times A$). 42
4.2 Transposed data matrix ($A \times I$). 43
4.3 Attribute $\times$ group matrix ($A \times G$). 43
4.4 Instance $\times$ group matrix ($I \times G$). 44
4.5 Instance $\times$ Group matrix ($I \times G$) for the example introduced in Chapter 4. 46

5.1 Summary of the datasets. 50
5.2 P-values, average number of rules for CHI-BD with *Summarizer* and its percentage of decrease when compared to CHI-BD without *Summarizer* . 52
5.3 Gmeans for CHI-BD with and without *Summarizer*. 54
5.4 Gmeans for CHI-BD with and without *Summarizer* 55

7.1 Number of rules generated by CHI-BD for each of the tested approaches and for each dataset. 67

LIST OF ACRONYM

FRBCS	Fuzzy Rule-Based Classification System	15
FS	Fuzzy Systems	7
KB	Knowledge Base	14
DB	Data Base	14
RB	Rule Base	14
FCM	Fuzzy C-means	30
HDFS	Hadoop Distributed File System	22
ML	Machine Learning	18
PCA	Principal Component Analysis	30
PC	Principal Component	35
LSA	Latent Semantic Analysis	30

Chapter 1

INTRODUCTION

A trending topic that has gained attention in the business and scientific environments is the phenomenon called “big data”. This term is used to refer to the huge amount of data that is generated every day, which has promoted/forced changes in traditional data mining and Machine Learning (ML) approaches. Big data includes structured and unstructured data that can come from a variety of sources and formats. For example, bank transactions, mobile activities, data in the format of video, text, image, and so on.

Among the wide range of challenges that arise with Big Data are classification tasks, where knowledge is extracted from data to predict future patterns. An expressive part of Big Data is in the format of text that can be used to solve various real life problems, such as spam detection (CHEN et al., 2015), author identification (SILVA; SILVA, 2017), web pages classification (SARAÇ; ÖZEL, 2014) and sentiment analysis (MEDHAT; HASSAN; KORASHY, 2014). Besides the increasing number of instances and attributes present in Big Data, uncertainty, vagueness, and noise inherent to such data can mask the valorous information to be obtained from Big Data, making it more difficult to perform classification tasks. The concept of uncertainty, for example, can be intuitively understood with the sentence “It is supposed to snow today”. It is uncertain because it might not snow all day. The phrase “This summer will be much hotter this year” exemplifies the concept of vagueness, since the vague expression “much hotter” indicates the lack of precision. Noise, on the other hand, is unwanted data examples or features that are not useful to understand the pattern of the data. These three phenomenons complicate data mining and ML processes.

More data usually lead to more accurate and precise information. However, in the big data scenario, information cannot be easily extracted by traditional data mining techniques. That is, standard data mining techniques and ML algorithms, such as the Chi et al.'s algorithm (CHI; YAN; PHAM, 1996), normally fail to scale up to huge amounts of data. This way, when using such standard data mining techniques for high dimensional datasets, analysis results are usually associated with poor performance of algorithms, decreasing of accuracy and recall, high complexity of the target functions, and model overfitting. Thus, to overcome the curse of dimensionality and big data, new techniques must be developed and/or standard techniques must be redesigned and adapted to the big data scenario.

On the other hand, Fuzzy Rule-Based classification Systems (FRBCS) are popular tools used to solve classification and pattern recognition problems (ISHIBUCHI; NAKASHIMA; NII, 2006). These tools can, very effectively, deal with the uncertainty, vagueness and noise inherent to data. For example, Chi et al.'s algorithm, that is a traditional FRBCS (CHI; YAN; PHAM, 1996), is a very effective rule based algorithm that works by turning data examples into rules where attributes of each data example are components of those rules. However, traditional FRBCSs, such as the above mentioned Chi et al.'s algorithm, were not designed to deal with the new challenges that come with big data problems, including scalability and noise accumulation. For instance, an impractical amount of time might be needed for training a ML model, as the Chi et al.'s algorithm, since much more data examples and attributes might be considered in the process. Besides the increasing time needed for processing such high dimensional datasets, complexity is also significantly increased. In the case of Chi et al.'s algorithm, complexity grows because of the increased number of rules, as well as the increased size of each rule. To address this difficulty, a lot of work have been done to adapt these techniques to big data (LÓPEZ et al., 2015; ELKANO et al., 2017a; RIO et al., 2015).

One of the biggest advantage of the Chi et al.'s algorithm is that it can be easily processed in parallel environments. That is, each data example can be turned into a rule independently from the other data points. This characteristic of the model facilitates the adaptation of the algorithm to the Big Data scenario. The adapted FRBCS techniques attempt to deal with large amounts of data by making use of the most popular paradigm for addressing big data: MapReduce (DEAN; GHEMAWAT, 2008). Shortly, MapReduce is a distributed programming model that splits the data processing in two stages: Map and Reduce. In the Map stage, the data is divided into multiple blocks and distributed among processing nodes that process each block of data independently. In the Reduce stage, the results coming from every node in the Map stage is aggregated and a final result is returned.

The first adaptation of the original Chi et al.'s algorithm using the MapReduce paradigm was proposed by López et al. (2015). The basic idea of the proposed approach, the Chi-FRBCS-BigDataCs algorithm, is to split the data into multiple training sets that will be processed independently by different nodes in the Map function. The results are then aggregated in the Reduce stage forming the final set of rules.

Later on, two different approaches were proposed by Rio et al. (2015) in order to generate better rules. The proposed algorithms, named Chi-FRBCS-BigData-Ave and Chi-FRBCS-BigData-Max, have the same Map stage as the Chi-FRBCS-BigDataCs. The algorithms differ in the Reduce stage, where Chi-FRBCS-BigData-Ave and Chi-FRBCS-BigData-Max algorithms have more sophisticated methods for aggregating the rules and generating the final result.

The latest adaptation of Chi et al.'s algorithm was proposed by Elkano et al. (2017a). The proposed method, Chi-BD, is more robust and aims to eliminate the dependency of the model on the distribution of the training data in each mapper. Chi-BD computes the rule weights considering the whole dataset instead of dividing the data into multiple training sets. This way, the authors defend that rules quality is not affected by the degree of parallelism. The rule generation process is more complex for the Chi-BD algorithm if compared to the algorithms mentioned above. However, Elkano et al. (2014a) show that Chi-BD outperforms Chi-FRBCS-BigdataCS in terms of runtime and classification performance when dealing with big data problems.

The existing adaptations of FRBCSs have shown that applications of fuzzy rule-based systems are promising alternatives to deal with big data problems. However, the methods presented above leave gaps that motivated the development of this work. Such gaps and the motivation of this work are presented in Section 1.1.

1.1 MOTIVATION

The algorithms mentioned above are adaptations of the traditional Chi et al.'s algorithm for fuzzy rule bases generation in the context of big data. The literature has shown that the MapReduce paradigm enables the use of fuzzy rules to solve big data problems. The existing methods were intensively studied in datasets with vertical high dimensionality (high number of instances) and showed to be very effective. Nevertheless, the performance of the algorithms in datasets with an increasing number of attributes (horizontal high dimensionality) as well as an increasing number of instances was not explored.

Moreover, the adapted methods presented above were tested and validated in datasets with millions of instances, but the larger dataset had a maximum of 54 attributes. It is well known that some classification tasks, such as text classification, may have datasets with thousands or millions of attributes as well as thousands of instances. Text datasets are examples of data that can have huge amounts of attributes since every unique word across all text data points is a different attribute. In such cases, only distributing data in multiple mappers may not overcome the high dimensionality problem since each mapper will have a reduced number of instances but will still have a very high number of attributes. In other words, vertical high dimensionality is addressed by distributing data across multiple mappers, but horizontal high dimensionality will still be a problem in each mapper.

One way to overcome the horizontal high dimensionality problem is to perform some feature selection or dimensionality reduction techniques before running classification algorithms (DENG et al., 2018; CHANDRASHEKAR; SAHIN, 2014). Feature selection techniques select the "most important" subset of attributes following some specific criteria (LI et al., 2017; LABANI et al., 2018). Despite these techniques be widely used for solving classification problems, there are some concerns regarding to the loss of information when discarding some attributes. Feature selection processes might not guarantee a fair representation of all classes by the selected features, especially for imbalanced datasets. These techniques evaluate each attribute individually, and dependencies between the attributes might be ignored, resulting in the selection of redundant attributes and on the discard of relevant ones (CHANDRASHEKAR; SAHIN, 2014; KUMBHAR; MALI, 2016).

Considering that FRBCSs were not still tested in datasets with both vertical and horizontal high dimensionality, and considering that feature selection and dimensionality reduction techniques may discard useful information, this work was developed concerning to the following hypothesis and objectives.

1.2 HYPOTHESIS AND OBJECTIVES

In the Big Data scenario, there may be datasets with a high number of attributes and a high number of instances, horizontal and vertical high dimensionality respectively. The existing MapReduce approaches for FRBCSs are focused on datasets with vertical high dimensionality, leaving a gap on problems with both horizontal and vertical high dimensionality.

In this work, we deal with the aforementioned drawbacks by clustering the attributes of such datasets to reduce horizontal high dimensionality. Consequently, classification performance may be improved by using such groups of attributes as the new feature space. It is expected that the smaller feature space will represent the data properly and that the classification algorithms will present good classification results in terms of accuracy. It is also expected that a smaller number of rules will be generated since a smaller number of features will be used, which will result in more interpretable systems. Therefore, with our approach, a smaller feature space is built for each dataset using the well-known fuzzy clustering algorithm, Fuzzy C-means (FCM) (BEZDEK, 1981). In addition to FCM, two established dimensionality reduction techniques were applied: Principal Component Analysis (PCA) and Latent Semantic Analysis (LSA). Both approaches use Singular Value Decomposition (SVD) and leverage the idea that meaning can be extracted from context. In LSA, the context is extracted using the term×document matrix, while in PCA the context is provided through the term covariance matrix. In this sense, the current work was performed under the following hypothesis:

It is possible to reduce dimensionality, without discarding information, and build better models by using groups of attributes in the classification process in a fuzzy rule-based classification system using the MapReduce paradigm.

To test the hypothesis, the goal of this work was to use FCM, PCA, and LSA to reduce the dimensionality of the feature space before applying a FRBCS under the MapReduce paradigm. It is expected that a smaller number of rules be generated and that the models present better accuracy.

1.3 WORKFLOW

To achieve these goals, this dissertation is organized as follows.

Chapter 2 - Fuzzy Systems. In this chapter, we present the main topics of fuzzy systems including fuzzy sets, fuzzy rules, fuzzy rule-based classification systems and fuzzy inference system.

Chapter 3 - Horizontal Dimensionality Reduction. In this chapter, we present the basic concepts of the Fuzzy C-Means algorithm and two other well-known dimensionality reduction techniques: Principal Component Analysis (PCA), and Latent Semantic Analysis (LSA).

Chapter 4 - Big Data. In this chapter, it is given an introduction to the Big Data concepts as well as a description of the main tools to deal with Big Data problems.

Chapter 5 - Related Work. In this chapter, it is given an overview on what has already been done for fuzzy rule based classification systems on the context of Big Data.

Chapter 6 - Proposal. In this chapter, it is given a deeper evaluation of the existing adaptations of the Chi et al.'s algorithm to the Big Data scenario. The developed work is also explained in details.

Chapter 2

FUZZY SYSTEMS

Fuzzy Systems (FS) are systems that make use of the fuzzy sets theory (ZADEH, 1965). Fuzzy sets can handle uncertainty, ambiguity or vagueness inherent to data in a very effective manner. Instead of dealing with crisp logic, fuzzy sets allow objects to belong to many sets with different membership degrees.

In the next sections, the main concepts about fuzzy sets, operation on fuzzy sets, fuzzy relations and membership functions will be discussed.

2.1 FUZZY SETS

Traditional set theory uses probability theory to deal with crisp events. In the crisp scenario, an element can either belong to a set or not belong to it. For instance, considering an article as an element and two sets representing 'politics' and 'sports', in a crisp approach, the article can either be about politics or sports. It is not considered the possibility of the article being about politics and sports at the same time. On the other hand, fuzzy sets theory can generalize the traditional set theory by providing a mechanism that allow different membership degrees of an example to every set. In this case, the article could belong to the politics set with a 0.6 membership degree, and it could belong to the sports set with a 0.4 membership degree, for example.

The definition of a fuzzy set goes from a characteristic function $\{0,1\}$, where 1 and 0 indicates if the element belongs to the set or not, to a membership function $[0,1]$, where the membership degree is indicated instead. Details of membership functions are given in the next section.

2.1.1 Membership functions

The degree to which an event belong to a set can be established by using a membership function. The notation of a membership function of a fuzzy set A for the set X of elements is defined below.

$$A : X \to [0, 1]$$

$A(x)$ associates each element $x \in X$ to a degree of membership in the interval $[0, 1]$ to the fuzzy set A (KLIR; YUAN, 1995). A given element may belong to different fuzzy sets with different degrees. This way, membership functions can overlap with each others.

Piecewise linear are the most popular membership functions because of their simplicity. One of the most popular, and the one that will be focused in this work, is the triangular membership function, shown below.

$$A(x, a, m, b) = \begin{cases} 0, & \text{for} \quad x \leq a, \\ \frac{x-a}{m-a}, & \text{for} \quad x \in (a, m), \\ 1, & \text{for} \quad x = m, \\ \frac{b-x}{b-m}, & \text{for} \quad x \in (m, b), \\ 0, & \text{for} \quad x \geq b. \end{cases}$$

for $a \leq m \leq b$. The graphical representation of the triangular membership function is shown in Figure 2.1.

2.1.2 Operation on fuzzy sets

Operations over fuzzy sets are extensions of operations over crisp sets. The basic connective operators of the standard fuzzy set operations are *intersection*, *union* and *complement*. The fuzzy complement applied to the fuzzy set A with membership function $A(x)$, and the fuzzy intersection and union operators applied to the fuzzy sets A and B with membership functions $A(x)$ and $B(y)$ can be defined as follows:

- Fuzzy complement
$$\bar{A}(x) = 1 - A(x), \quad x \in X \tag{2.1}$$

- Fuzzy intersection

$$A \cap B(x) = min_{(x \in X)}\{A(x), B(x)\}, \quad x \in E \tag{2.2}$$

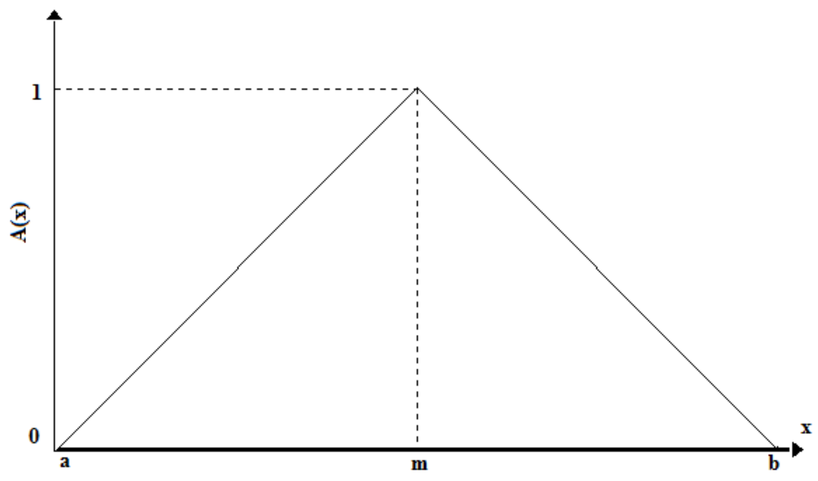

Figure 2.1 Triangular membership function for a fuzzy set.

- Fuzzy union

$$(A \cup B)(x) = max_{(x \in X)}\{A(x), B(x)\}, \quad x \in E \tag{2.3}$$

The generalized fuzzy intersection, also known as t-norms, are binary operations $t : [0,1] \times [0,1] \rightarrow [0,1]$ that follow the following properties.

Commutativity: $atb = bta$
Associativity: $at(btc) = atb(tc)$
Monotonicity: if $b \leq c$, then $atb \leq atc$
Absorption: $at0 = 0$
Neutrality: $at1 = a$

For $a, b, c \in [0,1]$. Some examples of t-norms are:

Standard intersection:$atb = min(a, b)$
Algebraic product: $atb = ab$
Algebraic difference: $atb = max(0, a + b - 1)$
Boundary conditions:

$$atb = \begin{cases} b, & if \quad a = 1 \\ a, & if \quad b = 1 \\ 0, & otherwise \end{cases}$$

The same way, the generalized fuzzy union, also known as t-conorms or s-norms, are binary operations $t : [0, 1] \times [0, 1] \rightarrow [0, 1]$ that follow the following properties.

Commutativity: $asb = bsa$
Associativity: $as(bsc) = asb(sc)$
Monotonicity: if $b \leq c$, then $asb \leq asc$
Absorption: $as1 = 1$
Neutrality: $as0 = 0$

For $a, b, c \in [0, 1]$. Some examples of s-norms are:

Standard union: $xasb = max(a, b)$
Limited sum: $asb = min(1, a + b)$
Algebraic sum: $asb = a + b - a \times b$
Boundary conditions:

$$asb = \begin{cases} a, & if \quad b = 0 \\ b, & if \quad a = 0 \\ 1, & otherwise \end{cases}$$

2.1.3 Fuzzy Relations

Fuzzy relations can be generalized from the traditional set concepts. They represent the degree of association between elements of two or more fuzzy sets (NICOLETTI; CAMARGO, 2004). A binary fuzzy relation $R(D, X)$ over two fuzzy sets, D and X, is a fuzzy set defined by the membership function $R(d, x)$. $R(d, x)$ defines the degree of association between d and x through the Cartesian product in 2.4.

$$R : D, X \rightarrow [0, 1] \tag{2.4}$$

The matrix representation of the relation between the instances $D = \{d_1, d_2, ...d_n\}$ and the attributes $X = \{x_1, x_2, x3, ...x_m\}$ is the following:

$$R = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \vdots & \dots & \dots & \vdots \\ r_{n1} & r_{n2} & \dots & r_{nm} \end{bmatrix}$$

in which $r_{ij} = R(d_i, x_j)$ for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, m$.

The membership functions taken from the union and intersection of two binary fuzzy relations, R_1 and R_2, are shown in 2.5 and in 2.6, respectively.

$$R(d, x) = R_1(d, x) s R_2(d, x), \quad \forall d, x \in D \times X \tag{2.5}$$

$$R(d, x) = R_1(d, x) t R_2(d, x), \quad \forall d, x \in D \times X \tag{2.6}$$

Where t is a t-norm, s is a s-norm, and R_1 and R_2 are two fuzzy relations defined in the same space $D \times X$.

Fuzzy relations can be combined through the *fuzzy composition* operation. The composition of two fuzzy relations $R_1(x, y)$ and $R_2(y, z)$ defined in the Cartesian products $X \times Y$ and $Y \times Z$, respectively is represented in 2.7 and 2.8.

$$R(\langle x, z \rangle) = (R_1 \circ R_2)(\langle x, z \rangle) = max_{y \in Y} t[R_1(\langle x, y \rangle), R_2(\langle y, z \rangle)] \tag{2.7}$$

$$R(\langle x, z \rangle) = (R_1 \circ R_2)(\langle x, z \rangle) = min_{y \in Y} s[R_1(\langle x, y \rangle), R_2(\langle y, z \rangle)] \tag{2.8}$$

where t is a t-norm and s is a s-norm.

The above mentioned definitions are important for the understanding of fuzzy rules. In the next sections, it will be discussed the concepts of linguistic variables, fuzzy rules and fuzzy inference.

2.2 COMPUTATION WITH FUZZY RULES

In fuzzy logic, fuzzy rules have been elected as key engines for expressing knowledge and for understanding the vagueness and ambiguity inherent to data (DUBOIS; PRADE, 1996). All the computational steps involving rules apply mechanisms that use fuzzy rules to make inference. Rules are often expressed as linguistic variables, explained in the next section.

2.2.1 Linguistic Variables

Linguistic variables make it possible for knowledge to be mathematically represented. This representation enables ambiguous problems to be manipulated by computers. Linguistic variables are variables expressed in a natural language, instead of numerical values. For example, the temperature of a room is usually qualified as “hot” and “cold” in real life. A linguistic variable “Temperature” can be decomposed, for example, with the linguistic terms: *too-cold, cold, warm, hot, too-hot*, as a categorical variable. Values of a linguistic variable are words or sentences. The representation of a fuzzy decomposition of the fuzzy variable Temperature is shown in Figure 2.2.

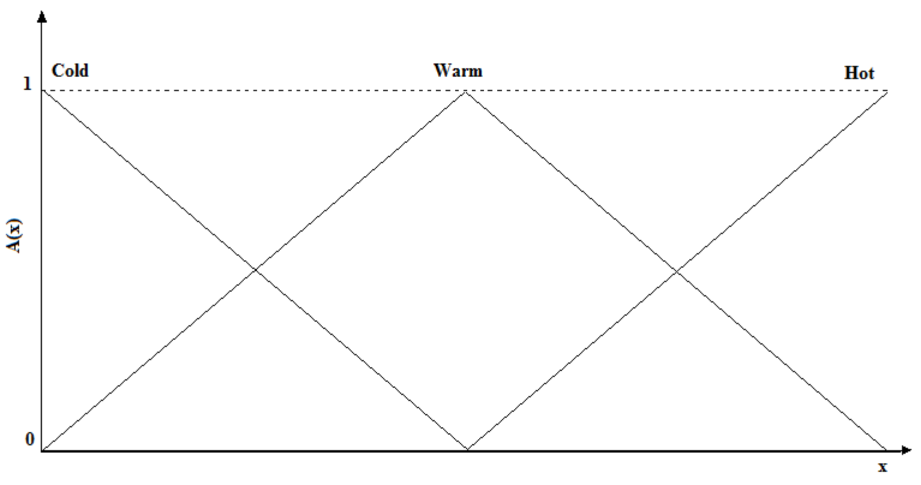

Figure 2.2 Membership function for the linguistic variable “Temperature”.

Each term of this decomposition can cover an interval of the overall values of the temperature. This way, a value of temperature can be represented by multiple terms at the same time with different membership degrees. On the graph, x is representing the temperature, and A(x) is the probability associated with it. A temperature of 10 degrees Celsius, for example, can have a low membership degree for warm and hot, and a high membership degree for cold.

In the next section, it will be presented a discussion over the main topics about fuzzy rules and fuzzy inference.

2.2.2 Fuzzy Rules

Using rules for knowledge representation is an old technique that is still widely used nowadays. Knowledge representation through the use of fuzzy rules is very popular due to the simplicity of the linguistic representation. Fuzzy rules are IF-THEN rules expressed by the linguistic terms of the Linguistic variables. For example, the Temperature variable can build a conditioner system through the use of rules as follows:

IF (temperature is *cold* OR *too-cold*) THEN action IS *turn heat on*
IF (temperature is *hot* OR *too-hot*) THEN action IS *turn heat off*
IF (temperature is *warm*) THEN action is *no-change*

The rules described above are fuzzy conditional statements. They associate a condition, described using linguistic terms and fuzzy sets, to an output or conclusion. The collection of fuzzy conditional statements forms the rule-base or the rule set of a fuzzy system.

The main concepts about fuzzy rules were presented above, and more detailed information can be found in (ZADEH, 1965). In order to draw conclusions from a set of rules, fuzzy inference must be aplied. The main concepts about fuzzy inference will be presented in the next section.

2.2.3 Fuzzy Inference

Fuzzy sets itself cannot lead to useful and practical decisions. Fuzzy inference processes need to be applied in order to draw conclusions from a set of rules. In the fuzzy inference process, membership functions are combined with the control rules and a fuzzy output is derived. The most basic inference rule for fuzzy systems is the compositional rule of inference. The basic case where there is one variable in the antecedent and one variable in the consequent will be presented here.

Let A be a fuzzy set over X and B a fuzzy set over Y. The fuzzy relation $R(x, y)$ over X and Y is inducted from the rule IF X is A THEN Y is B. This relation can be defined by the function:

$$R(x, y) = f(A(x), B(y))$$

f can be a fuzzy conjunction, disjunction or implication (detailed in (PEDRYCZ; GOMIDE, 1998)).

In order to use the fuzzy rule above to make an inference, it is necessary to know same information about the antecedent variables. For example, lets say that X is A', then it can be concluded that Y is B', in which B' is defined by Equation 2.9.

$$B'(y) = \sup_{x \in X}[A'(x) t R(x, y)] \tag{2.9}$$

where sup is the supremum operator and t is a t-norm. The application of the compositional rule of inference can be easily extended to the case where there are multiple variables. The general process of computation using fuzzy rules is based on the compositional rule of inference.

In the next section, the main concepts over FS will be presented.

2.3 FUZZY SYSTEMS

Among the different available Fuzzy Systems, the **FRBCSs!** (**FRBCSs!**) are the focus of this work. The main characteristics of these systems will be described in the next sections.

2.3.1 Fuzzy Rule-Based Classification Systems

The **FRBCSs!** are composed of two main components: Knowledge Base (KB) and Inference system. The method produces an interpretable model with human readable linguistic labels. The KB is composed of the Data Base (DB) and the Rule Base (RB), as shown in Figure 2.3. The DB contains the membership functions of the attributes, and the RB contains the set of fuzzy rules that describe the problem. The inference system will later derive conclusions from the rules.

To implement a FRBCS, the following steps are needed:

- *Fuzzification*: converts crisp sets into fuzzy sets by deriving the membership functions.
- *Fuzzy Inference Process*: combines the membership functions with the control rules to derive individual results for each rule.
- *Defuzzification*: converts the fuzzy output from the previous step back to a crisp output. This step is not always necessary, depending on the goal of the FS.

The fuzzification step usually involves two process: deriving the membership functions for the input and output variables and representing them with linguistic terms, as dis-

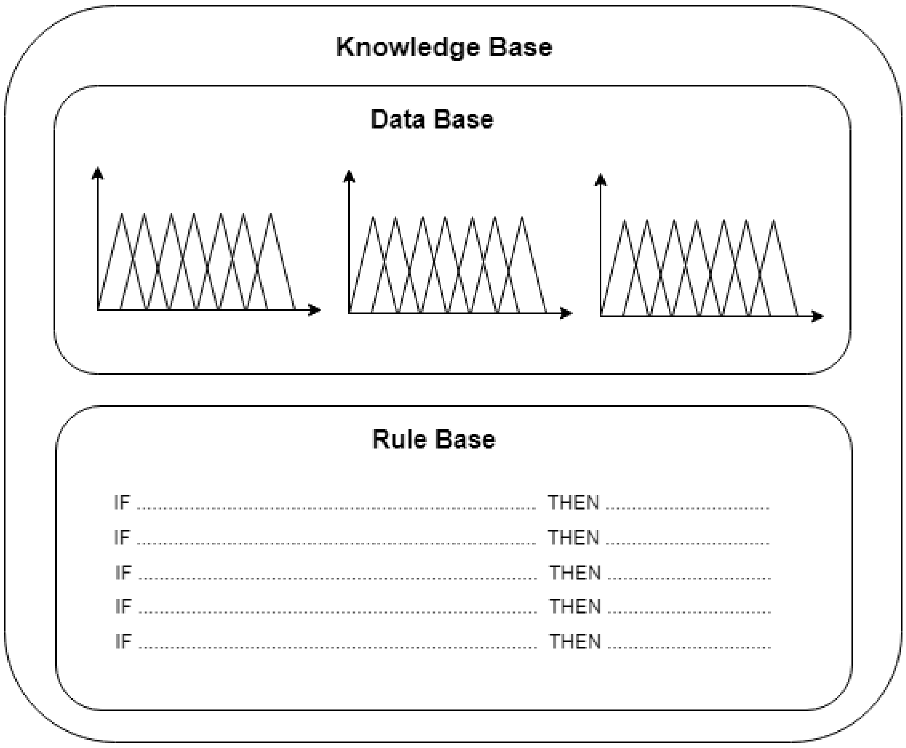
Knowledge Base
Data Base
Rule Base
IF .... THEN ....
IF .... THEN ....
IF .... THEN ....
IF .... THEN ....
IF .... THEN ....

Figure 2.3 Representation of a KB of a Fuzzy Rule-Based Classification System (FRBCS).

cussed earlier. In other words, these processes are equivalent to transforming a classical set into a fuzzy set with varying degrees.

In the fuzzy inference step, conclusions are derived from the combination of the input, output membership functions and the fuzzy rules. The way the fuzzy sets obtained in the inference process are used depends on the type of the system and the problem being solved. The output from the FS can be either turned into linguistic or numeric values (defuzzification).

Shortly, a fuzzy process is a crisp-fuzzy-crisp process where the initial input and the final output must be crisp values, but a fuzzy inference process must be performed in the middle of the process.

2.3.2 Fuzzy Inference System

There are two types of fuzzy inference that are widely used in many different fields: Mamdani fuzzy inference, (MAMDANI, 1976), and Sugeno fuzzy inference, (TAKAGI; SUGENO, 1985). The difference between the fuzzy models lie in the consequent of the rules.

The Mamdani systems have fuzzy propositions in the antecedent and consequent of the rules. On the other hand, Sugeno systems have propositions in the antecedent of the rules, but the consequent of the rules is a function of the antecedent part. A representation of both fuzzy models is shown below.

Mamdani:
IF X_1 is A_{i1} AND ... X_n is A_{in} THEN Y IS B

Sugeno:
IF X_1 is A_{i1} AND ... X_n is A_{in} THEN $f(x_1, \ldots, x_n)$

where $X_1, \ldots, X_n$ are the input linguistic variables, Y is an output linguistic variable, $A_{i1}, \ldots, A_{in}$ and B are linguistic terms, and f is a function applied over the input values $x_1, \ldots, x_n$.

The Mamdani system outputs a fuzzy set that is then defuzzified in order to obtain a precise numerical value as a result. The Sugeno system already outputs a precise numerical value. This way, the defuzzification step it is not necessary.

Since the goal of this work is to use fuzzy rules to classificate documents, a classification step is necessary after the rules generation process is finished. The fuzzy rule-based classification process is explained in the next section.

2.3.3 Fuzzy Classification

In a fuzzy rule-based classification system, the goal is to classify instances, represented by m attributes, into a set of predefined classes. Rules in a FRBCS follow the general pattern of the rules presented in Section 2.2.2 and can be represented as follows:

IF X_1 is A_{i1} AND ... X_m is A_{im} THEN $Class = C_c$

where $X_1, \ldots, X_m$ are attributes represented by linguistic variables, $A_{i1}, \ldots, A_{im}$ are the linguistic terms representing each attribute, and C_c is the class that the specific instance belongs to.

Fuzzy methods for classification are inference procedures that take *if...then* rules and instances with attribute values in order to determine the class of such instances. The most used methods for fuzzy classification are the Classical Fuzzy Reasoning and the General Fuzzy Reasoning.

The Classical Fuzzy Reasoning method seeks to classify instances according to the rules with the higher compatibility degrees. Consider $d = \{a_1, s_2, \ldots, d_m\}$ an instance to be classified, and $R_1, R2, \ldots, R_S$ a set of S rules from the classification system, each of them with m antecedents. Consider $A_i(a_i)$, $i = 1, \ldots, m$, the membership degree of the attribute a_i to the i-th fuzzy set of the rule R_k. The Classical Fuzzy Reasoning method classifies the instance d according to the following steps.

- Calculating the compatibility degree between the instance d and each rule R_k, $k = 1, \ldots, S$.
  $Compat(R_k, d) = t(A_1(a_1), A2(a_2), \ldots, A_m(a_m))$, where t is a t-norm.
- Finding the rule with the higher compatibility degree with the instance d.
  $maxCompat(R_k, d), k = 1, 2, \ldots, S$
- Defining the class C_j to the instance d, where C_j is the consequent of the rule R_k that has the higher compatibility degree with instance d.

The General Fuzzy Reasoning method combines information from all rules regarding to a specific class in order to classify an instance d. The steps are described as follows.

- Calculating the compatibility degree between the instance d and each rule R_k, $k = 1, \ldots, S$.
  $Compat(R_k, d) = t(A_1(a_1), A2(a_2), \ldots, A_m(a_m))$, where t is a t-norm.
- Calculating, the classification degree, $Class_c$, of d for each class C. This is possible by aggregating the compatibility degrees of all rules whose predict class is C.
  $Class_C = f\{Compat(R_k, d)\}|\{CistheclassofR_k\}$, where f is an aggregating operator such that $min \leq f \leq max$.
- d will be classified into class C, where C is the class corresponding the the maximum value of $Class_c$ from the previous step.

Both techniques uses the KB built previously to identify the class that most associates to the new instance. The difference between the techniques is in the way the calculation of the association degree between the instance and each rule/class is performed.

2.4 FINAL CONSIDERATIONS

In this chapter, the main concepts of a FRBCS were described with great attention given to the approaches that were used in this work. The next chapter is devoted to explain the main concepts related to Big Data and, main tools used nowadays to deal with it, as well as the most recent adaptations of traditional Machine Learning (ML) algorithms to the Big Data scenario.

Chapter

3

VERTICAL DIMENSIONALITY REDUCTION

As discussed earlier, vertical high dimensionality refers to the huge amount of instances, while horizontal high dimensionality refers to the huge amount of attributes. Although Big Data can be related to both horizontal and vertical high dimensionalities, Big Data is usually approached as vertical high dimensional datasets. CHI-BD and most of the cited adaptations discussed here make use of MapReduce and Hadoop for dealing with vertical high dimensionality.

In this chapter, it is given a general description of Big Data and the challenges that arises with it for traditional ML techniques to extract useful knowledge from such big amounts of data. The MapReduce paradigm and the Hadoop framework are usually used to tackle high dimensionality problems and are also briefly discussed here. In addition to those concepts, it is also given a general idea of existing adaptations of traditional ML algorithms to the Big Data scenario.

Finally, it is also given a report on the existing FRBCSs adapted to the Big Data scenario. All the existing adaptations are based on Chi et al.'s algorithm, so the mentioned algorithm is also briefly discussed. CHI-BD, that is the latest adaptation of the Chi et al.'s algorithm to Big Data, is discussed in more detail since it will be used for evaluating the feasibility of this work.

3.1 BIG DATA

Despite its first use happened years before, the "Big Data" term was officialized in 2013 when Oxford English Dictionary introduced it for the first time in its dictionary. According to the dictionary, the definition of the term "Big Data" is:

"Extreme large datasets that may be analyzed computationally to reveal patterns, trends, and associations..."

As the definition presented above says, the high dimensionality inherent to Big Data has brought much interest because of the possibility to improve data analysis and knowledge extraction. Nevertheless, Big Data has also raised many concerns regarding to the difficulty to deal with the large amount of data related to it. Big Data can be so large and complex that traditional software tools or data processing applications are not capable of processing or analyzing (FERNÁNDEZ et al., 2014).

In order to describe the challenges associated with Big Data, six defining properties are commonly used: volume, velocity, variety, variability, veracity, and value. Each of them is shortly described below:

- **Volume**: refers to the huge amount of data that need to be processed in order to get useful information.
- **Velocity**: refers to the speed of data processing that has to be done in a reasonable period of time.
- **Variety**: refers to the number of types of data that can be structured or unstructured, such as pictures, audios, text among others.
- **Variability**: refers to the inconsistency that can be associated to data. It defines the need to get useful data even in conditions of high unpredictability.
- **Veracity**: refers to the reliability of the captured data. Biases, noises and abnormalities are present in Big Data, and a good data cleansing has to be assured in order to only store valuable data.
- **Value**: refers to the value of such data to the purpose of the analysis.

These Big Data problems/characteristics are present in a vast number of fields. For example, Instagram generates 95 million photos and videos each day, 45,788 trips are taken by Uber riders every minute, 156 million emails are sent every minute, and so on. To address these problems and extract useful knowledge from data, several solutions have been proposed recently. Most of the proposed solutions regarding to classification problems combines the MapReduce model with the Apache Hadoop computing framework (LOPEZ et al., 2014; FERNÁNDEZ et al., 2017; MAILLO; TRIGUERO; HERRERA, 2015a; PRIYADARSHINI et al., 2015a; RÍO et al., 2014).

It is worth mentioning that the works cited above use MapReduce and Hadoop to tackle vertical high dimensionality, which is generally refered to as Big Data. The MapReduce model and the Appache Hadoop computing framework are described in the next sections.

3.2 MAPREDUCE

MapReduce is a distributed programming model proposed by Google in 2004 (DEAN; GHEMAWAT, 2008) with the goal of simplifying the distributing process. Distributed programming is a strategy that distributes the computation flow along large-scale clusters of machines. Here, clusters of machines refers to a set of connected computers that work together to achieve results faster. The term should not be confused to clusters of data generated by the clustering algorithms, also referred to as groups. The MapReduce model divides the computational flow in two main stages: Map and Reduce. Each stage is organized around key-value pairs. The Map and Reduce functions are described below.

Map function: in this stage, the data is automatically divided into independent data blocks and distributed along storage nodes. Nodes are computing units that can be used for storage or processing. The data is analyzed independently by each node, and, for each input, intermediate results are returned by each node.

Reduce function: the Reduce function collects and aggregates the results returned by each node in the Map function and produces a final result.

The two biggest advantages of MapReduce are: parallel processing and data locality. When using a MapReduce approach, one is dividing the job among multiple nodes (slave nodes), and each slave node executes part of the job simultaneously. This way, MapReduce is a Divide and Conquer approach where the big problem is divided into smaller sub-problems and each sub-problem is solved independently. The solutions of the sub-problems are finally merged and the final solution of the original problem is gotten. This approach helps to get the final solution much faster since multiple resources are working on solving the problem at the same time. A graphic representation of how the work can be divided into multiple computing units is shown in Figure 3.1.

Figure 3.1 illustrates a distributed computing approach where 4 slave nodes are available for working in parallel. A master node is responsible for distributing the work between the slave nodes. The second advantage of MapReduce, data locality, refers to the fact that the processing unit is moved to the data, instead of data being moved to the processing unit. As shown in Figure 3.1, data is distributed among multiple nodes

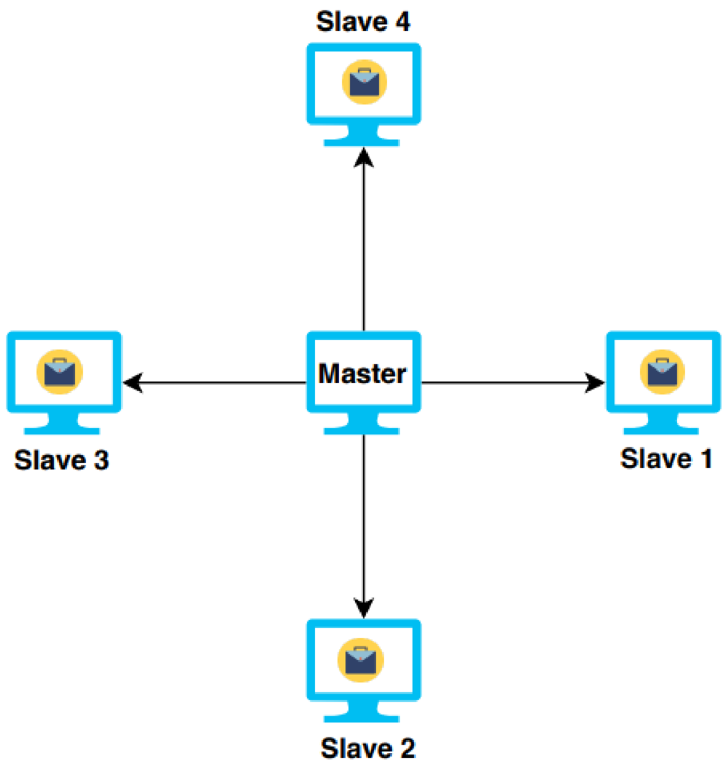

Figure 3.1 MapReduce work split between computing nodes.

where each node stores and processes only the part of the data residing on it. The most relevant open source implementation for the MapReduce programming model is Appache Hadoop, described in the next section.

3.3 APACHE HADOOP

Apache Hadoop (Hadoop) is an open source framework for Big Data processing and storage in a distributed environment across *clusters* of computers (WHITE, 2012). The Hadoop framework implements the MapReduce model over a distributed file system called Hadoop Distributed File System (HDFS). HDFS replicates the data files in many storage nodes, which facilitates data transfer among nodes and allows the system to keep working even when one or several nodes fail. The MapReduce scheme allows the processing of the data in each node.

Hadoop has a high capability for data storage, data management, and data processing on a petabyte scale. It will be considered in this work due to its open source nature and facilities regarding to its installation and use.

In order to better understand the process, a Hadoop MapReduce processing Model is presented in Figure 3.2.

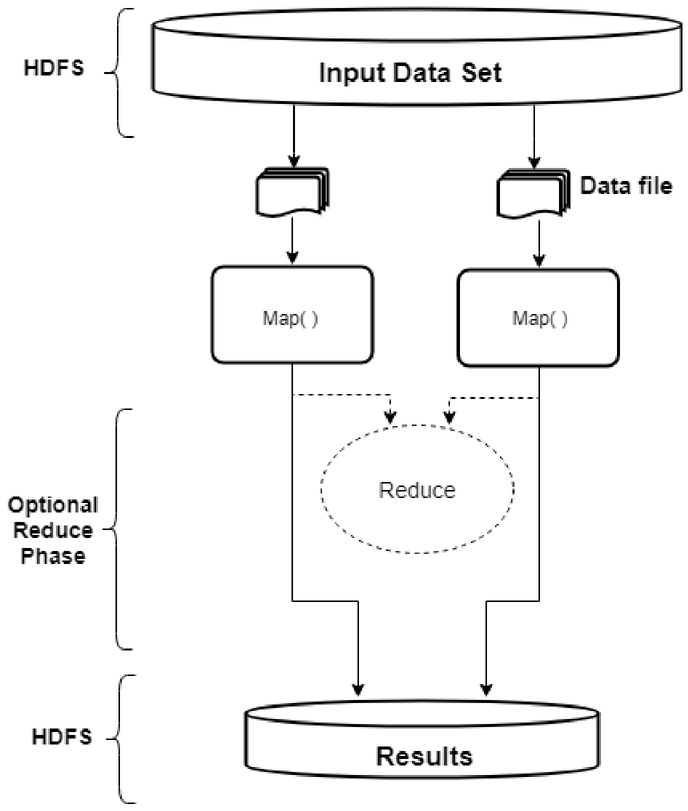

Figure 3.2 Representation of a Hadoop MapReduce processing Model.

Figure 3.2 pictures the general process of a Hadoop MapReduce Model. The process can be adapted to the user MapReduce scheme.

In the next sections, we present a discussion about the existing ML models adapted for Big Data problems. Most of articles discussed in the next sections make use of the MapReduce paradigm and the Hadoop framework for dealing with vertical high dimensionality. Great attention was given to the works about FRBCSs, since it is the focus of this work.

3.4 ADAPTATIONS OF TRADITIONAL MACHINE LEARNING ALGORITHMS FOR BIG DATA

Big Data has promoted/forced changes in traditional ML methods and tools in order to address exponentially growing processing times as the number of examples increases.

In the last few years, researchers have studied the implementation of those traditional ML algorithms in a MapReduce framework (CHU et al., 2007). It has been found that MapReduce programming models allow the user to speed up the training process of a variety of ML algorithms. Algorithms for learning fuzzy rules, for example, are very well suited for parallel environments since rules creation can be adapted to a map and reduce stage. Examples of ML algorithms already implemented in a MapReduce framework include: Random Forests (LI et al., 2012; WAKAYAMA et al., 2015; HAN; LIU; SUN, 2013), Distributed Decision Trees (DAI; JI, 2014), Fuzzy Decision Trees (SEGATORI; MARCELLONI; PEDRYCZ, 2018), K-Nearest Neighbors (MAILLO; TRIGUERO; HERRERA, 2015b; ANCHALIA; ROY, 2014; ZHANG; LI; JESTES, 2012), Support Vector Machine (SUN; FOX, 2012; KHAIRNAR; KINIKAR, 2015), Fuzzy rules (LOPEZ et al., 2014; ELKANO et al., 2017b, 2017a; RIO et al., 2015; FERNÁNDEZ et al., 2017), among many others.

Most of the algorithms cited above were adapted to parallel environments for dealing with vertical high dimensionality. Cases where both vertical and horizontal high dimensionality are present on the data, as it can be the case of text datasets, were not explored by these algorithms. A brief discussion about some of the distributed implementations cited above will be held here in order to emphasize the lack of approaches to deal with horizontal high dimensionality co-occurring with vertical high dimensionality. It will be given great attention to FRBCSs since it is the main focus of this work.

Decision tree learning algorithms have been proposed for managing Big Data by adopting the MapReduce paradigm (WANG et al., 2015; DAI; JI, 2014). Parallel implementations of Random Forests have also been proposed, and it was reported great improvements in terms of computational costs (LI et al., 2012; WAKAYAMA et al., 2015; HAN; LIU; SUN, 2013). In (RÍO et al., 2014), combinations of different techniques do deal with imbalanced Big Data, such as ROS (Random Oversampling), RUS (Random Undersampling), the SMOTE (Synthetic Minority Oversampling Technique) algorithm and cost-sensitive learning were adapted to the MapReduce scheme and combined with the Random Forest (RF) algorithm. Although it was shown gain in processing time, parallel implementation of random forests requires great care since each tree of the forest is built based on only a small subset of the data. This way, each tree may not represent the whole dataset, and the final model can easily over-fit the training data. In this study, the main concern was vertical high dimensionality where tests were made with up to 11 million examples but only with a maximum of 41 features.

Other than decision trees and random forest algorithms, the K-Nearest Neighbor (k-

NN) algorithm is also well known in data mining because of its simplicity and effectiveness. Parallel implementations of the method have already been proposed in (MAILLO; TRIGUERO; HERRERA, 2015b) and in (MAILLO et al., 2017). In (MAILLO; TRIGUERO; HERRERA, 2015b) a Hadoop implementation of MapReduce for K-NN (MR-KNN) was proposed, and the results showed a great improvement in the processing time while maintaining the classification rate as the original k-NN model. The algorithm was tested in a dataset with a high number of examples but only 10 features. This leads to a lack of comprehension about the behavior of the algorithm under high dimensionality both ways: examples and features. This scenario may lead to other problems since the KNN is a memory intensive algorithm.

MR-KNN inspired the implementation of the algorithm under Apache Spark, leading the authors to propose an iterative MapReduce-based approach for k-NN algorithm implemented under Apache Spark (kNN-IS) in (MAILLO et al., 2017). The results showed improvements since kNN-IS allowed the reduction of the runtime by almost 10 times in comparison to MR-KNN while maintaining the classification rate. The memory consumption issue was addressed by the authors that defend that kNN-IS calculates the solution with more than one iteration by splitting the test set when the memory capacity of the cluster exceeds. Tests were made with horizontally larger datasets (up to 631 features), but the classification rate was only discussed for the smaller ones (up to 18 features).

Support vector machine methods have also been intensively studied in the MapReduce scenario. It was shown that an iterative MapReduce implementation of SVM is efficient for solving Big Data problems (SUN; FOX, 2012). Some initial directions can be found in the following references (KIRAN et al., 2013; KHAIRNAR; KINIKAR, 2015; SUN; FOX, 2012; PRIYADARSHINI et al., 2015b).

The above mentioned algorithms are just some of the broad amount of methods implemented in the MapReduce scenario. The discussion held above shows how MapReduce approaches have been extensively used for adapting ML algorithms to Big Data analysis. It can also be observed that vertical high dimensionality is the main focus of the approaches. Datasets with both vertical and horizontal high dimensionality are not approached by the adaptations shown above. Implementation of many other algorithms can be found in the literature and will not be discussed here since the main focus of this work is FRBCSs. These types of systems were considered in this work since they are well know for dealing with the uncertainty, vagueness and noise inherent to data.

3.5 FUZZY RULE BASED CLASSIFICATION SYSTEMS FOR BIG DATA

FRBCSs are popular tools to solve classification problems that have been explored in many different fields (SANZ et al., 2014, 2015; NAKASHIMA et al., 2007). These tools are well known for obtaining satisfactory accuracy results and for being able to produce interpretable models through the usage of linguistic labels. However, standard FRBCSs were not primarily designed to process large datasets and its performance is affected by the high number of instances and/or attributes associated with Big Data.

In order to overcome this issue, (LÓPEZ et al., 2015) proposed the first adaptation of the original Chi algorithm (CHI; YAN; PHAM, 1996) to Big Data scenarios: Chi-FRBCS-BigDataCS. The basic idea of the algorithm is to use Map and Reduce techniques to process data in separate groups and create sets of rules that are then aggregated. The algorithm splits the training data into multiple training sets distributed among the mappers. Each mapper builds its rule base using the Chi et al.'s algorithm with the associated training set. All rule bases are then aggregated and a final model is generated. If there are duplicated rules, the one with the highest weight is kept.

Although this approach overcomes the Big Data problem in the sense of decreasing the processing time, the results are sensitive to the degree of parallelism. That is, the more mappers are added to the process, the poorer rule weights may be obtained since they depend on the quality of the sample in each mapper.

Different alternatives have been proposed to obtain better rule weights or to better aggregate the rules in the reduce stage (ELKANO et al., 2017a; RIO et al., 2015). In (RIO et al., 2015), the Chi-FRBCS-BigData algorithm was proposed with two variations: Chi-FRBCS-BigData-Max and Chi-FRBCS-BigData-Ave. Both variations share the same initial procedure and the Map function. The algorithms differ on the Reduce stage, where each algorithm has a different approach for choosing the rules that have the same antecedents and different consequents. Chi-FRBCS-BigData-Max deals with these cases by choosing the rules with the higher rule weight. Chi-FRBCS-BigData-Ave, on the other hand, calculates the average rule weight for each case, and chooses the rule with the higher average rule weight.

The average alternative is more computational intensive since it calculates the average of each set of rules. Although Chi-FRBCS-BigData-Ave algorithm may take longer to run, it is less sensitive to the distribution of the samples in each mapper. This results in better classification rates compared to the Chi-FRBCS-BigData-Max algorithm. Both alternatives were tested and presented good results in datasets with vertical high dimensionality with millions of examples. However, the number of attributes of the tested databases were up to 54. This way, the behavior of the algorithm with vertical and horizontal high dimensionality is still unknown.

Even though Chi-FRBCS-BigData-Ave is less sensitive to samples distribution in each mapper, the resulting models will still be different if the configuration of the clusters changes. This causes the model to be less accurate as more mappers are added. With the goal of eliminating the dependency of the model on the distribution of the training data in the mappers, (ELKANO et al., 2017a) proposed a new approach to calculate rule weights: the CHI-BD model. This proposed technique computes rule weights considering the whole dataset instead of dividing the data into multiple training sets. This way, rules quality is not affected by the degree of parallelism. That is, the CHI-BD model is able to provide exactly the same model regardless of the number of mappers.

The workflow of the CHI-BD algorithm differs from the previous ones in the rule weights generation process. In the Map stage, the rules are generated by each mapper and the list of possible consequents for each antecedent is returned. No weights are taken into account in this stage. In the Reduce Stage, the whole rule base is loaded into each mapper and the matching degrees in all mappers are summed and used to compute the rule weights. Since the rule weights are calculated based on the whole Rule Base, the degree of parallelism does not affect the resulting model. (ELKANO et al., 2017a) show that the CHI-BD algorithm outperforms the Chi-FRBCS-BigdataCS in terms of runtime and classification performance when dealing with Big Data problems. The algorithm behaves well when dealing with datasets of millions of examples. However, the algorithm was only tested in datasets with up to 54 attributes.

CHI-BD is the most recent adaptation of Chi et al.'s algorithm to the Big Data scenario. Because of that, it was used in this work for assessing the feasibility of our approach. CHI-BD shows that FRBCSs can be easily implemented in parallel processing environments and are very effective for dealing with Big Data classification problems. Greater details about CHI-BD will be given in the next section.

3.5.1 CHI-BD

All the existing FRBCSs adapted to deal with Big Data following a MapReduce paradigm are based on Chi et al.'s algorithm, a traditional FRBCS learning method (CHI; YAN; PHAM, 1996). The flow of the traditional Chi et al.'s algorithm is presented below and its latest adaptation to Big Data, CHI-BD, is presented later.

Suppose we have n instances with m attributes classified into c classes. The Chi et al.'s rule generation process is the following:

1. A membership function is built for each of the m attributes.

2. Linguistic labels are built for each attribute through membership functions. The linguistic labels can be, for example, Low/Medium/High, Bad/Medium/Good, etc.

3. A fuzzy rule is generated for each example, being each rule based on the linguistic labels.

4. The membership degree of each instance to every fuzzy set is computed using the formulas presented in Chapter 2.

5. For every instance and for each attribute, the linguistic label with the greatest membership degree is selected.

6. The antecedent part of each instance is the intersection of the selected linguistic labels, and the consequent part is the instance class.

7. The rule weight is computed using some existing method.

At the end of the process, there will be n rules, one for each instance. To finalize the process, a cleaning step is performed where duplicated and meaningful rules are deleted. That is, rules with negative values are deleted, and rules that share the same antecedent part but have different consequent parts will be filtered, and the rule with the greatest weight will be kept.

The latest adaptation of the Chi et al.'s algorithm to the scenario of Big Data was proposed by (ELKANO et al., 2017b). The proposed algorithm, Chi-BD, is a more robust version of Chi et al.'s over the MapReduce scenario. The method is divided into two stages, where each stage is composed of a MapReduce process. The flow of the algorithm is presented below.

1. **Rules generation process**

    1.1. *Map stage*

    1.1.1. Data is split into different mappers

    1.1.2. In each mapper, one rule is created for each example

    1.2. *Reduce stage*

    1.2.1. Rules with the same antecedent are grouped and the list of all possible consequents is returned.

1.2.2. A new rule base is created with antecedents and the list of consequents.

2. **Computation of rule weights**

   2.1. *Map stage*

   2.1.1. Each mapper loads the rule base previously obtained.

   2.1.2. In each mapper, one rule is created for each example

   2.1.2.1. For each rule, the matching degrees of all examples related to each possible class of the rule is computed.

   2.2. *Reduce stage*

   2.2.1. The matching degrees in all mappers are summed and used to compute the rule weights.

As the flow above shows, the rule base generation process is very sophisticated and more complex than Chi et al.'s algorithms. The authors show that CHI-BD outperforms the previous adaptations of Chi et al.'s algorithm in terms of runtime and classification performance when dealing with Big Data problems.

Chi-BD is the most recent adaptation of Chi et al.'s algorithm to the Big Data scenario. The authors have shown that CHI-BD is very effective for processing datasets with vertical high dimensionality. Also, it is shown that fuzzy rule-based systems can be easily implemented in a parallel processing environment.

3.6 FINAL CONSIDERATIONS

In this chapter, the main concepts related to Big Data were presented. The MapReduce paradigm was described, and Hadoop, the main tool to deal with Big Data problems was briefly discussed.

In addition to Hadoop and MapReduce, it was given a brief presentation of the works that have been done for dealing with Big Datasets. The existing works show that the MapReduce scheme makes it possible to adapt traditional algorithms to the Big Data scenario. There are many other works that aim to adapt traditional techniques to Big Datasets, but they were not presented here since the main focus of this work is in the FRBCS.

Details about Chi et al.'s algorithm and its latest adaptation to the Big Data scenario were also given.

Since the technique developed in this work makes use of the Fuzzy C-Means algorithm, it is important to understand the main concepts related to clustering algorithms. On that behalf, the basic concepts of clustering algorithms will be presented in the next chapter. Great attention will be given to the Fuzzy C-Means algorithms.

Dimensionality Reduction

The proposed approach aims to reduce horizontal high dimensionality by creating groups of attributes as a new feature space of the dataset to be classified. Groups of attributes can be created by means of clustering algorithms. There are many different clustering techniques available in the literature (SAXENA et al., 2017). There are crisp and fuzzy techniques, and each technique suits specific situations and specific structures of data. Crisp techniques are those in which every instance belongs to only one group, such as the K-Means and K-Nearest Neighbors algorithms. On the other hand, fuzzy clustering algorithms allow each instance to belong to every group with different membership degrees. Fuzzy C-means (FCM) is the most popular fuzzy clustering algorithm and will be considered in this work due to its simplicity and effectiveness.

In addition to the FCM clustering technique, two established dimensionality reduction techniques were applied for reducing horizontal high dimensionality: Principal Component Analysis (PCA) and Latent Semantic Analysis (LSA). Both approaches use Singular Value Decomposition (SVD) in the process of reducing dimensionality and leverage the idea that meaning can be extracted from context. In LSA, context is extracted by means of the term$\times$document matrix, while in PCA context is provided through the term covariance matrix. Similar to the FCM clustering approach mentioned above, applying PCA or LSA to a horizontal high dimensional dataset will reduce the feature space without discarding attributes individually. These approaches reduce the risk of discarding important attributes and allow for considering a significantly smaller feature space.

The definition of clustering and the basic elements involved in the clustering process will be discussed in the next sections. Details about PCA and LSA are also given in the next sections.

3.7 CLUSTERING ALGORITHMS

Clustering algorithms are unsupervised techniques that aim to identify *clusters*/groups of similar instances (FAHAD et al., 2014). The instances to be grouped can be anything of interest: cars, neighborhoods, people, text, or even abstract concepts. The similarity between instances is taken from the attributes that characterize them. For example, the attribute “Model year” characterizes the object “car”. This way, older cars and newer cars tend to be in different groups, for example. In the clustering process, the goal is to group similar instances in the same group. Instances in different groups must be as much different as possible.

In a crisp approach, each instance belongs to only one specific group of similar instances, while in a fuzzy approach each instance belongs to every group with different membership degrees. In order to quantify the similarity between instances and define the most suitable groups for each instance, clear measurement of similarity and dissimilarity must be defined.

Distance and similarity are key concepts for constructing clustering algorithms, since groups are built based on similarity and distance measurements. A similarity function $Sim()$ computes similarity degrees between instances, whose values are in the range [0,1]. Consider a set of n examples $D = \{d_1, d_2, \ldots, d_n\}$, where d_k, $k = 1, \ldots, n$, is a m-dimensional point. Usually, if $Sim(d_i, d_j) > Sim(d_i, d_l)$, where i, j, $l \in k$, we can say that instance d_j is more similar to instance d_i if compared to instance d_l. The Cosine similarity measure is commonly used in the context of text processing due to the high dimensionality of the instances to be compared (STREHL; GHOSH; MOONEY, 2000). The Cosine similarity between two instances is defined as follows.

$$Sim(d_i, d_j) = cos(\theta) = \frac{\vec{d_i} \cdot \vec{d_j}}{||\vec{d_i}|| \cdot ||\vec{d_j}||} \tag{3.1}$$

where $\vec{d_i}$ and $\vec{d_j}$ are instances in the vector form and θ is the angle between the two vectors.

Other than defining similarity between the instances, a number of steps are needed to perform a clustering process. In a standard process of clustering, the following steps are needed:

- Data pre-processing: In this step, data is prepared to start the clustering process. The most representative features are selected, the attributes are transformed and normalized if necessary, and outliers are identified and treated.

- Similarity measure selection: In this step, the most appropriate similarity measure must be selected.

- Clustering algorithm execution: In this step, the chosen clustering algorithm is executed.

- Results evaluation: In this step, the quality of the results is evaluated. The parameters of the algorithm are tunned and the best number of groups is identified. There are statistical techniques that help in this process.

- Results interpretation: In this step, the conceptual meaning of each group is identified.

The steps listed above might be followed for general applications of clustering algorithms. Further details about the specific FCM algorithm, chosen for this work, are presented below.

3.7.1 Fuzzy C-Means

Fuzzy C-Means (FCM)(BEZDEK, 1981) is the most popular fuzzy clustering algorithm and was considered in this work due to its simplicity and effectiveness. As mentioned earlier, for FCM, an example can belong to all groups with different membership degrees. Membership degrees are associated with the distance of the examples to the centroids of the groups. The more distant is the example to the centroid of some group, the smaller is its membership degree concerning to that group.

A fuzzy clustering is composed by a set of c groups, denoted by $P = \{A_1, A_2, \ldots, A_c\}$, and a partition matrix $W = w_{k,p} \in [0, 1]$, for $p = 1, \ldots, c$, where each element $w_{k,p}$ represents the membership degree of the example k in the group A_p (KLIR; YUAN, 1995).

The sum of all membership degrees for a given example d_k must be equal to 1, as shown in Equation 3.2.

$$\sum_{p=1}^{c} w_{k,p} = 1 \tag{3.2}$$

Each group A_p must contain at least one example with non-zero membership degree and must not contain all the points with membership degrees equal to 1, as shown in Equation 3.3.

$$0 < \sum_{k=1}^{n} w_{k,p} < n \tag{3.3}$$

A brief description of the FCM algorithm is given below.

1. An initial partition matrix W must be selected.
2. The centroids of each group must be computed considering the fuzzy partition.
3. The fuzzy partition must be updated until the stopping criteria is achieved.

The stopping criteria can be: a specific number of iterations has been executed, the change in the fuzzy partition is below some threshold.

The goal of the FCM algorithm is to minimize the sum of the squared error (SSE), as shown in Equation 3.4. That is, the goal is to minimize the distances between the examples and the centroids c_p of the groups.

$$SSE = \sum_{k=1}^{n} \sum_{p=1}^{c} w_{k,p}^{f} dist(d_k, c_p)^2 \tag{3.4}$$

where $f > 1$ is the fuzzifier parameter that can influence the performance of the FCM algorithm.

In this work, FCM is being used to create groups of similar words, since the work focus on text datasets. To give an idea on how to create groups of words using FCM, four simple phrases were created with two clear definitions of groups: food and sports. FCM will be implemented with $c = 2$ groups and the membership degrees of each word to each group will be presented. The phrases are described below:

Table 3.1 Simple texts example

ID	Document
doc_1	'I like eating delicious food and eating sweet candy'
doc_2	'I just ate a delicious sweet banana'
doc_3	'My team loves playing footbal'
doc_4	'I love to play the footbal game'

The first column of Table 3.1 represents the id given to each document (phrase). We can note the phrases are very simple with the goal of facilitating the understanding and interpretation of the FCM results. The documents are represented in terms of words count in Table 3.2. Observe that the representation is already considering text pre-processing, that includes stemming and stop words removal.

Table 3.2 Words count representation

	banana	candy	delicious	eat	food	football	game	love	play	sweet	team
doc_1	0	1	1	2	1	0	0	0	0	1	0
doc_2	1	0	1	0	0	0	0	0	0	1	0
doc_3	0	0	0	0	0	1	0	1	1	0	1
doc_4	0	0	0	0	0	1	1	1	1	0	0

Since the goal is to create groups of words and not groups of documents, the matrix input for FCM is the transpose of 3.2, where rows are words and columns are documents.

A weighting criteria for each word should also be considered for better data representation. The transposed matrix with the TF-IDF weighting criteria is shown in Table 3.3.

Table 3.3 Transposed matrix with TF-IDF representation.

	doc_1	doc_2	doc_3	doc_4
banana	0	2	0	0
candy	2	0	0	0
delicious	1	1	0	0
eat	4	0	0	0
food	2	0	0	0
football	0	0	1	1
game	0	0	0	2
love	0	0	1	1
play	0	0	1	1
sweet	1	1	0	0
team	0	0	2	0

Table 3.3 has words as rows and documents as columns and is in the right format input for FCM. Note that, usually, matrix input for clustering algorithms is in the format of documents as rows and words as columns, since the goal is to put similar documents in the same groups. Here, the goal is to create groups of similar words instead of groups of similar documents. This way, words are put in rows and documents in columns. The membership degrees of each word to each group returned by FCM is shown in Table 3.4.

Table 3.4 Membership degrees generated by FCM.

	Group 1	Group 2
banana	0.4404	**0.5596**
candy	**0.9731**	0.0269
delicious	**0.7204**	0.2796
eat	**0.7905**	0.2095
food	**0.9731**	0.0269
sweet	**0.7204**	0.2796
footbal	0.0099	0.9901
game	0.2111	0.7889
love	0.0099	0.9901
play	0.0099	0.9901
team	0.2111	0.7889

As it can be seen in Table 3.4, two groups of words were created since the c parameter in FCM was set to 2. Words related to food are marked in bold along with its higher membership degrees. If a crisp approach were to be considered, a threshold (say 0.5) would be defined, and words with membership degrees above the threshold would be assigned to one specific group. However, since we are using a fuzzy approach, each word belongs to both groups with different membership degrees. For the word 'candy', for example, the membership degree to Group 1 was equal to 0.9731, while the membership degree to Group 2 was equal to 0.0269. That means that the word 'candy' is closer to words in Group 1 than to words in Group 2, but the word is still similar to words in Group 2 with a lower intensity.

Observing the words in general, it can be noted that most of the words related to food had a higher membership degree to Group 1. In the same way, all words related to sports had a higher membership degree to Group 2. From that, one can infer that Group 2 is represented by documents about sports while Group 1 is represented by documents about food.

The example presented above is a simplified explanation of how FCM algorithm would be applied to extract groups of words from text datasets. After generating groups of words, text documents would then be represented in terms of groups instead of being represented in terms of words directly. This allows for reducing feature space, which helps to solve the horizontal high dimensionality problem with Big Datasets. Details about how to proceed after generating groups of attributes will be presented in the next chapters.

Other than creating groups of attributes, in this work, horizontal high dimensionality reduction was also considered using PCA and LSA. Details about both approaches are given in the next sections.

3.8 PRINCIPAL COMPONENTS ANALYSIS (PCA)

PCA is a mathematical matrix decomposition technique for dimensionality reduction. It reduces dimensionality by finding linearly uncorrelated vectors, called Principal Component (PC), that minimize information loss and maximize the explained variance of the data. In other words, PCA is made through an orthogonal linear transformation that finds new coordinate systems in which the first coordinate carries the greatest variance, the second coordinate carries the second greatest variance, and so on. Each new coordinate is a PC. The goal of each PC is to carry as much variability as possible, so only a few PCs are capable of representing most of the variability of the data. It was first proposed by Karl Pearson (PEARSON, 1901) and has been widely used since then (JOLLIFFE; CADIMA, 2016).

In our example of food and sports documents, a total of 11 words are used to describe the data. By applying PCA to the raw data, a smaller subset of PCs would be enough to explain the data in exchange of the 11 words. As for the number of groups in FCM, the number of PCs to be considered on the data analysis is also defined by the user. The choice of the number of PCs can be made with the help of the percentage of variability explained by each PC.

The steps for performing PCA on a $n \times m$ dimensional dataset are: data standardization, covariance matrix computation, eigenvectors and eigenvalues computation, and feature vector construction. Descriptions of each step mentioned above will be given in the next sections.

3.8.1 Data Standardization

The absolute values of the attributes are taken into account when creating Principal Components. This way, variables that have higher absolute values will have more importance in each PC. For example, if one has attributes 'height' (in centimeters) and 'weight' (in kilograms) of a person and wants to build Principal Components, the attribute 'height' will have more importance to the PCs than the attribute 'weight' since, usually, the absolute values of heights are higher than the absolute values of weights. That is not a desirable behavior since one expects that PCs carry information about all the attributes, not only specific ones. That is, every attribute must contribute equally to the PCs definitions.

In order to prevent specific attributes to be more important than others only due to scale, data standardization prior to the PCs construction must be performed. That is, all m attributes must be transformed to the same scale before creating PCs. A simple data standardization technique would be to set every attribute to have mean 0 and standard deviation of $\sigma^2 = 1$, the often called z-score normalization. For a specific attribute M with data examples doc_{km}, with $k = 1, 2, ..., n$, this transformation can be done with the following formula (GARCÍA; LUENGO; HERRERA, 2015).

$$doc_{km} = \frac{doc_k - \bar{M}}{sd(M)} \tag{3.5}$$

where $\bar{M}$ is the mean of M and $sd(M)$ is the standard deviation of M. After applying the standardization to all attributes of the data set, data is prepared to the next steps.

3.8.2 Covariance Matrix Computation

The $m \times m$ covariance matrix is built to represent the relationship between all the attributes. Considering a set of attributes X, Y and Z, the covariance matrix would be represented as follows (GARCÍA; LUENGO; HERRERA, 2015).

$$Cov(X, Y, Z) = \Sigma = \begin{pmatrix} Cov(x, x) & Cov(x, y) & Cov(x, z) \\ Cov(y, x) & Cov(y, y) & cov(y, z) \\ Cov(z, x) & Cov(z, y) & Cov(z, z) \end{pmatrix}$$

The covariance matrix has the variances of each attribute on its diagonal. The covariances between the attributes represent the relationship between them. A positive covariance between X and Y indicates that the two variables increase or decrease together, that is, X and Y are correlated. On the other hand, a negative covariance between X and Y indicates that one increases when the other decreases, that is X and Y are inversely correlated. The covariance matrix, Σ, representing the relationship between the attributes can be calculated by a matrix multiplication, as shown below.

$$\Sigma = \frac{W^T W}{n - 1}$$

where W is the $n \times m$ data matrix representign the n attributes and the m examples. Principal Components are calculated through the covariance matrix of the attributes. This is done through eigenvectors and eigenvalues computation of the covariance matrix. The eigenvectors and eigenvalues calculation is discussed below.

3.8.3 Eigenvectors and Eigenvalues Computation

The m eigenvectors of the Covariance matrix are the directions of the axes where there is the most variance (most information), and those axes are called Principal Components. On the other hand, the eigenvalues of the covariance matrix are the coefficients attached to the eigenvectors, which give the amount of variance carried in each Principal Component.

The eigenvalues can sort the eigenvectors in descending order to provide a ranking of the most important Principal Components. The percentage of contribution of each PC to explaining the variability of the data is the ratio between the variance of that PC (associated eigenvalue) and the total variance, as shown below (PEARSON, 1901):

$$\frac{\lambda_j}{\sum_{j=1}^{m} \lambda_j}$$

Where λ_j is the eigenvalue associated to the j-th PC. The eighen values are in such a way that $\lambda_1 > \lambda_2 > ... > \lambda_m$, which reflects the fact that the first PCs carries the most variability of the data. The typical goal of PCA is to keep only the first most important PCs. That characterizes the last step of PCA, described in the next section.

3.8.4 Feature Vector Construction

After computing the PCs and its eigenvalues, one should define the appropriate number of Principal Components to keep for further analysis. The $c \ll m$ Principal Components are selected, and the $n \times c$ final data set is represented by the PCs. In order to choose the appropriate amount of PCs, one can make use of the plot of the eigenvalues of each PC. One example of such plot is given in Figure 3.3.

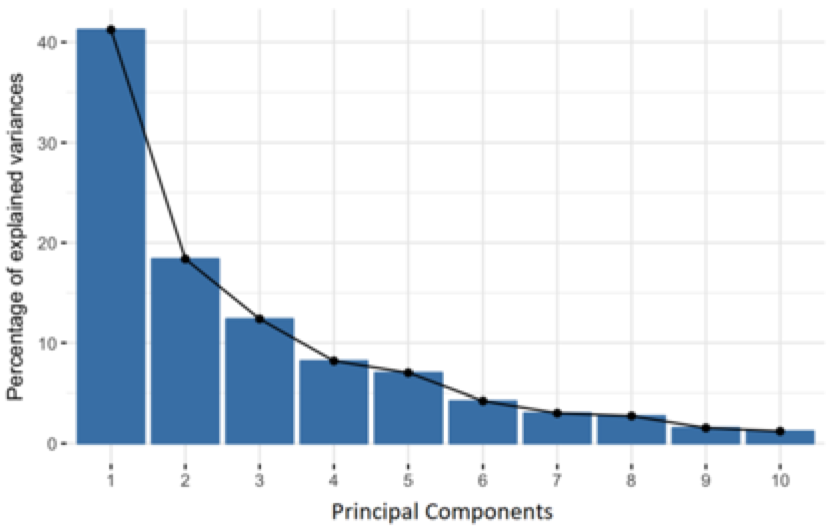

Figure 3.3 Percentage of explained variance by each Principal Component.

As it can be seen in Figure 3.3, the first Pirncipal Component carries most of the variability of the data. On the other hand, the last PC explains very little variance. There is no such perfect number of PCs appropriate for representing the original data set. This way, one should consider the trade-off between explained variability and dimensionality of the data set. In Figure 3.3, for example, the last few components carry less than 5% of variability of the data and may be excluded from the analysis with no much loss of information.

3.9 LATENT SEMANTIC ANALYSIS (LSA)

LSA is a technique usually used for natural language processing to analyze relationships between terms and documents (LANDAUER; FOLTZ; LAHAM, 1998). Its main idea is to filter noise and reduce dimensionality by finding the smallest set of concepts that explain all the documents, where concepts are patterns of words that usually appear together in documents.

Similar to PCA, LSA decomposes the data matrix into singularvectors and singular-values and defines a smaller feature space to explain the variability of the data. The main difference between the techniques is that LSA decomposes the original data matrix instead of the covariance matrix. LSA is commonly used for understanding concepts hidden in text datasets, as well as for reducing dimensionality when the original data set is presumed too large. Another common use of LSA is for reducing sparsity, which is a characteristic of matrices in which most of the elements are zero. That is a very common characteristic of text datasets where there are as many columns as unique words, and every word column is filled with zero when the word is not present in the text document.

In the example presented for explaining FCM, the input for LSA would be the matrix shown in Table 3.3. LSA takes the term$\times$document matrix and applies the Singular Value Decomposition (SVD) technique to obtain the reduced data space (GOLUB; REINSCH, 1971). SVD is a factorization of a $n \times m$ matrix, D, into three components USV^T. Where U is an $n \times c$ orthogonal matrix, whose columns are defined by the left-singular vectors of D. S is a $c \times c$ diagonal matrix, V is a $n \times c$ matrix, with V^T being the transpose of V. The columns of V are defined by the right singular vectors of D. The value $c \ll m$ is called the rank and defines the number of singular values that are to be kept.

Similar to PCA, the number of singular values in the diagonal matrix S defines the amount of variance explained by each of the singular vectors, and it is used to define the new reduced dimension of the transformed data matrix. As in PCA, after applying LSA to the original data matrix, the appropriate number of dimensions should be chosen to represent the original data set. Once the reduced feature space is selected, further analysis can be performed.

3.10 FINAL CONSIDERATIONS

In this chapter, the basic concepts of clustering algorithms were presented, as well as the basic concepts of the dimensionality reduction techniques PCA and LSA. Among the existing clustering techniques, the FCM algorithm was studied more deeply since it was chosen for implementing the proposed approach in this work.

As it has already been mentioned, this work aims to deal with big datasets and aims to use the MapReduce scheme to solve problems related to it. The MapReduce paradigm will be used after the horizontal dimensionality reduction techniques be performed on the datasets (FCM, PCA, and LSA). In the next chapter, the development of *Summarizer*, our proposed approach to reduce horizontal dimensionality by using groups of attributes is presented.

Chapter 4

SUMMARIZER

In the Big Data scenario, there may be datasets with vertical and horizontal high dimensionality (high number of attributes and high number of instances, consecutively). Existing MapReduce approaches are focused on datasets with vertical high dimensionality, leaving a gap on problems with co-occurring vertical and horizontal high dimensionality.

Although adaptations of the Chi et al.'s algorithm were proposed for a Big Data scenario, the MapReduce approach presented by recent algorithms deals only with datasets with vertical high dimensionality. However, in text datasets for example, the number of attributes is usually very high as well. In such cases, horizontal high dimensions of the datasets are not dealt with by the MapReduce approach proposed by the models.

To deal with the above drawbacks, the approach of this work consists in adding a grouping step to the FRBCS execution. In the new grouping step, a clustering algorithm is applied to the attributes before creating membership functions. This way, the new attributes of the proposed approach are groups of attributes, and the instances are represented by linguistic labels regarding those groups. Consequently, classification performance may be improved by using such groups of attributes as the new feature space.

It is expected that the smaller feature space represents the data properly and that the classification algorithms present good classification results in terms of accuracy. It is also expected that a smaller number of rules be generated since a smaller number of features is used, which results in more interpretable systems. Therefore, with our approach, a smaller feature space is built for each dataset using the well-known fuzzy clustering algorithm, Fuzzy C-Means (FCM)(BEZDEK, 1981).

In summary, the goal of the proposed approach is to reduce horizontal dimensionality by using groups of attributes to build membership functions in a FRBCS under the MapReduce paradigm. CHi-BD, the latest adaptation of FRBCS , which makes use of the MapReduce paradigm, was tested and compared with and without adding the grouping step.

The general idea of our work includes adding a clustering step before the classification process, by which the attributes of a dataset are groups of attributes obtained through the FCM algorithm and two other well-known dimensionality reduction techniques: PCA and LSA.

By reducing the high number of attributes to a much smaller number of groups, the problem of horizontal high dimensionality will no longer exist. This way, existing ML techniques, as CHI-BD for example, will then be suitable to solve classification problems. By considering this approach, the number of membership functions in the FRBCS will be much smaller since each membership function will represent a group of attributes instead of representing each attribute itself. By consequence, the number of rules of the classification system will also smaller and the rules will be much simpler.

For better understanding the *Summarizer* approach, consider a dataset with n instances and m attributes. In the case of text datasets, instances are documents and attributes are unique words present in the documents. The data matrix would be built as in Table 4.1.

Table 4.1 Data matrix of Instances and Attributes ($I \times A$).

Instance	**Attribute 1**	**Attribute 2**	$\dots$	**Attribute m**
Instance 1	d_{11}	d_{12}	$\dots$	d_{1m}
Instance 2	d_{21}	d_{22}	$\dots$	d_{2m}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Instance n	d_{n1}	d_{n2}	$\dots$	d_{nm}

where d_{ij} is the value of the i-th instance and the j-th attribute, with $i = 1, \dots, n$ and $j = 1, \dots, m$. The first step for creating groups of attributes is to transpose the data matrix, so an attribute $\times$ instance ($A \times I$) matrix is obtained. Here, we expect that a data preprocessing has already been applied. Since we are dealing with text datasets in this work, here we expect that common text pre-processing techniques has already been done, as for example words stemming, tokenization, and stop-words removal. For PCA, there is no need to perform a further transformation on the data. Table 4.1 is already in the appropriate input format for the PCA model. For FCM and LSA, further data transformation is necessary. The format of the transposed matrix, a necessary step for FCM and LSA, is presented in Table 4.2.

Table 4.2 Transposed data matrix ($A \times I$).

Attribute	**Instance 1**	**Instance 2**	$\ldots$	**Instance** n
Attribute 1	d_{11}	d_{21}	$\ldots$	d_{1n}
Attribute 2	d_{12}	d_{22}	$\ldots$	d_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Attribute m	d_{m1}	d_{m2}	$\ldots$	d_{mn}

Note that Table 4.2 has attributes as rows and instances as columns. In the case of text datasets, words would be in rows and documents in columns. The next step of the process is to apply FCM or LSA on the attributes with varying quantity of groups. The output in this step will be an attribute $\times$ group ($A \times G$) matrix, represented in Table 4.3. Note that the term 'Group' is being used through all the process, but it refers to semantic concepts when LSA is being used instead of FCM.

Table 4.3 Attribute $\times$ group matrix ($A \times G$).

Attribute	**Group 1**	**Group 2**	$\ldots$	**Group** c
Attribute 1	w_{11}	w_{12}	$\ldots$	w_{1c}
Attribute 2	w_{21}	w_{22}	$\ldots$	w_{2c}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Attribute m	w_{m1}	w_{m2}	$\ldots$	w_{mc}

In the case of FCM, w_{jl} represents the degree of membership of the attribute j to the group l, with $j = 1, \ldots, m$ and $l = 1, \ldots, c$, where $c \ll m$. Since the goal is to associate instances to groups or semantic concepts, matrix $A \times G$ will be converted to an instance $\times$ group matrix ($I \times G$). In order to quantify the relationship between instances and groups and build the $I \times G$ matrix, relationships between attributes and instances should be taken into account.

In the case of LCA, instead of the membership degrees, one would have values of the documents in respect to each LSA semantic concept. Since the output of LSA already gives a relationship between documents and semantic concepts, a conversion step is not necessary. In the same way, for PCA, a relationship between instances and Principal Components is already given as output of the technique. This way, a conversion step is not necessary.

There could be many ways of reflecting on the instances the relationship between attributes and groups created by FCM. This association is straight forward since every instance is associated to all attributes and every attribute is associated to all groups. *Summarizer* proposes a simple and effective way of taking these associations into account: a weighted average where weights are values of the attributes to each group. This way, the $I \times G$ matrix is created by calculating the weighted average between the values of the attributes in every instance and the membership degree of the attributes in the groups. The x_{il} component of the $I \times G$ matrix can be calculated as in Equation 4.1.

$$x_{il} = \frac{\sum_{j=1}^{m} d_{ij} \times w_{jl}}{\sum_{j=1}^{m} w_{jl}} \tag{4.1}$$

where $i = 1, \ldots, n$, $l = 1, \ldots, c$, and $j = 1, \ldots, m$. By calculating the weighted average of words in a document in respect to the value of that word in each group, we assure that the strength of each group in the specific document will be well represented. This way, documents whose words are closer to Group 1, for example, will have higher x_{il} values for Group 1. Matrix $I \times G$ is represented in Table 4.4.

Table 4.4 Instance $\times$ group matrix ($I \times G$).

Instance	**Group 1**	**Group 2**	$\ldots$	**Group** c
Instance 1	x_{11}	x_{12}	$\ldots$	x_{1c}
Instance 2	x_{21}	x_{22}	$\ldots$	x_{2c}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Instance n	x_{n1}	x_{n2}	$\ldots$	x_{nc}

Table 4.4 is the data in the final format to be used for classification tasks with the appropriate algorithms. As in any clustering approach, the number m of groups or semantic concepts should be chosen by the user. The final number of groups can be much smaller than the original number of attributes, making the horizontal high dimensionality a solved problem. The effectiveness of the approach was tested with CHI-BD, which deal with the vertical high dimensionality now that *Summarizer* has already tackled the horizontal high dimensionality problem.

In order to clarify the whole process that composes *Summarizer*, the above mentioned steps and matrices transformation involving FCM are summarized below. For PCA, it is not necessary to perform any transformation on the data format before or after the application of the technique. For LSA, only step 1 is necessary, since the input for the algorithm should be a A × G matrix and the output is already a I × G matrix (or Instance × semantic concept matrix, in this case).

Steps for performing *Summarizer* with the FCM approach

1. Transposing the original data matrix (instance × attribute) in order to get an attribute × instance matrix. This step is only necessary to FCM and LSA, since PCA takes as input the original I × A matrix.

   $$I \times A \rightarrow A \times I$$

2. Cluster the attributes with different number of groups. For FCM, the output in this step will be an attribute × group matrix. For LSA, the output is already the final format with associations between instances and semantic concepts. This way, when using the LSA approach, one can skip to step 4.

   $$A \times I \rightarrow A \times G$$

3. Calculating the weighted averages for each instance in each group to get the instance × group matrix. This step is only necessary for FCM since PCA and LSA already returns a instance × groups association.

   $$A \times G \rightarrow I \times G$$

4. Perform classification task using groups (FCM), PCs (PCA) or semantic concepts (LSA) as attributes.

To better explain how the whole process of *Summarizer* is done, consider the food and sports example introduced in Chapter 4. Table 3.2 is equivalent to the Instance × Attributes (I × A) data matrix of *Summarizer*, shown in Table 4.1. On the other hand, Table 3.3 is the transposed data matrix (A × I) equivalent to Table 4.2 of *Summarizer*. In the same way, Table 3.4 represents the Attribute × Group (A × G) matrix of the *Summarizer* approach. To obtain the Instance × Group (I × G) matrix, that is the output of *Summarizer*, Equation 4.1 should be applied using the data from I × A and A × G, on Tables 3.3 and 3.4 respectively. The calculation process for some of the values and the results after the calculations are shown below.

$$x_{11} = \frac{0{*}0.44+2{*}0.97+1{*}0.72+4{*}0.79+2{*}0.97+0{*}0.72+0{*}0.01+0{*}0.21+0{*}0.01+1{*}0.01+0{*}0.21}{0.44+0.97+0.72+0.79+0.97+0.72+0.01+0.21+0.01+0.01+0.21} = 1.6757$$

$$x_{12} = \frac{0{*}0.56+2{*}0.03+1{*}0.28+4{*}0.21+2{*}0.03+0{*}0.99+0{*}0.79+0{*}0.99+0{*}0.99+1{*}0.28+0{*}0.79}{0.56+0.03+0.28+0.21+0.03+0.99+0.79+0.99+0.99+0.28+0.79} = 0.2537$$

$$x_{21} = \frac{2*0.44+0*0.97+1*0.72+0*0.79+0*0.97+0*0.01+0*0.21+0*0.01+0*0.01+1*0.72+0*0.21}{0.44+0.97+0.72+0.79+0.97+0.72+0.01+0.21+0.01+0.01+0.21} = 0.4573$$

$$x_{22} = \frac{2*0.56+0*0.03+1*0.28+0*0.21+0*0.03+0*0.99+0*0.79+0*0.99+0*0.99+1*0.28+0*0.79}{0.56+0.03+0.28+0.21+0.03+0.99+0.79+0.99+0.99+0.28+0.79} = 0.2830$$

$$x_{31} = \frac{0*0.44+0*0.97+0*0.72+0*0.79+0*0.97+1*0.01+0*0.21+1*0.01+1*0.01+0*0.72+2*0.21}{0.44+0.97+0.72+0.79+0.97+0.72+0.01+0.21+0.01+0.01+0.21} = 0.0891$$

$$x_{32} = \frac{0*0.56+0*0.03+0*0.28+0*0.21+0*0.03+1*0.99+0*0.79+1*0.99+1*0.99+0*0.28+2*0.79}{0.56+0.03+0.28+0.21+0.03+0.99+0.79+0.99+0.99+0.28+0.79} = 0.7669$$

$$x_{41} = \frac{0*0.44+0*0.97+0*0.72+0*0.79+0*0.97+1*0.01+2*0.21+1*0.01+1*0.01+0*0.72+0*0.21}{0.44+0.97+0.72+0.79+0.97+0.72+0.01+0.21+0.01+0.01+0.21} = 0.0891$$

$$x_{42} = \frac{0*0.44+0*0.97+0*0.72+0*0.79+0*0.97+1*0.01+2*0.21+1*0.01+1*0.01+0*0.72+0*0.21}{0.56+0.03+0.28+0.21+0.03+0.99+0.79+0.99+0.99+0.28+0.79} = 0.7669$$

Refreshing from Chapter 4, Group 1 is represented by words about food (doc$_1$ and doc$_2$), while Group 2 is represented by words about sports (doc$_3$ and doc$_4$). The supposed meaning of each group could be inferred by the membership degrees of the words to each group. Most of the words about food had higher membership degrees to Group 1, indicating that words about food are closer together in Group 1. In the same way, all of the words about sports had higher membership degrees to Group 2, indicating that words about sports are closer together in Group 2.

Table 4.5 makes the link from words × groups to documents × groups. The calculated values shown in Table 4.5 reflects the distributions of the words in the groups, identified in Table 3.4. Two examples on how Equation 4.1 would be applied for obtaining the values on Table 4.5 were also show above (calculation steps for x_{11} and x_{12}). The relationship between words and groups are not enough for the data analysis since classification is performed over documents. This way, a relation between documents and groups should be defined. That is what Equation 4.1 of *Summarizer* does. It translates the relationships between words and groups to relationships between documents and groups.

Table 4.5 Instance × Group matrix ($I \times G$) for the example introduced in Chapter 4.

	Group 1	Group 2
doc$_1$	$x_{11} = 1.6757$	$x_{12} = 0.2537$
doc$_2$	$x_{21} = 0.4579$	$x_{22} = 0.2830$
doc$_3$	$x_{31} = 0.0891$	$x_{31} = 0.7669$
doc$_4$	$x_{41} = 0.0891$	$x_{41} = 0.7669$

Because doc$_1$ and doc$_2$ are the documents related to food, and words about food had higher membership degrees to Group 1, values for doc$_1$ and doc$_2$ on Group 1 are higher than values for doc$_1$ and doc$_2$ on Group 2. The same happens for doc$_3$ and doc$_4$. Since doc$_3$ and doc$_4$ are documents related to sports, and words about sports had higher membership degrees to Group 2, values for doc$_3$ and doc$_4$ on Group 2 are higher than values for doc$_3$ and doc$_4$ Group 1. These comparisons between groups meaning and documents classes (food and sports) show that a coherent association between groups of words and documents is made.

FCM was taken to present an example of *Summarizer* calculations, and membership degrees were used during the whole process. However, output values from PCA and LSA could be used the same way. The only difference is that PCA and LSA already returns the relationship between Principal Components and semantic concepts, for PCA and LSA respectivelly. This way, the whole conversion steps applied to FCM would not be necessary.

Summarizer can be used in any classification system for any type of data, since its main idea is to reduce the feature space of a dataset without loss of information. To check its feasibility when Big Data is considered, in this work, the experiments were performed for text classification tasks using the CHI-BD algorithm. Therefore, the experiments were performed as shown in the workflow in Figure 4.1: in Step 1, by means of FCM, PCA, and LSA, an horizontal dimensionality reduction is performed; and in Step 2, by means of CHI-BD, a vertical dimensionality reduction is performed for a classification task.

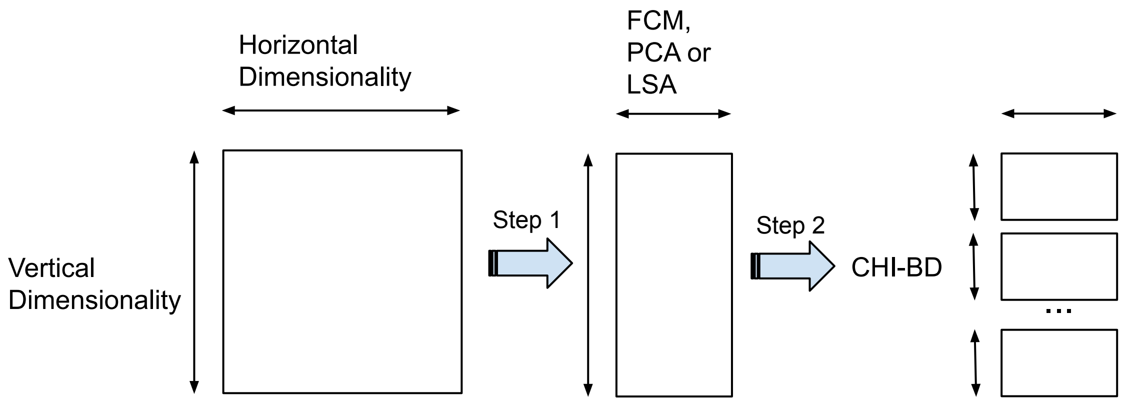

Figure 4.1 Workflow of *Summarizer*

The results obtained from our experiments are presented in the next chapter.

Chapter

5

EXPERIMENTS

This study is aimed at assessing the performance of our approach (*Summarizer*) by analyzing the results obtained in terms of classification accuracy and number of rules. We compare the results obtained through FCM, PCA and LSA algorithms to the results of CHI-BD with no clustering step. Section 6.1 describes the datasets, the parameters, and the statistical tests considered for the study. In sections 6.2 and 6.3 we analyze the results in terms of number of rules and classification performance.

5.1 EXPERIMENTAL FRAMEWORK

To conduct the experiments, five high dimensional datasets were selected from the LABIC website[1]. In order to increase the search space of our analyzes, we have obtained 9 binary classification problems by turning the original multi-class into multiple binary One-vs-All datasets. To do so, we selected a positive class and considered the rest of the classes as the negative class. Descriptions of the datasets are shown in Table 5.1 with the number of examples (#Examples), the number of examples of majority and minority classes (#maj;#min), and the number of attributes (#Attributes).

The experiments were conducted applying a 5-fold cross validation. Therefore, the result of each dataset with each technique was computed as the average of the accuracies in each fold.

The parameters considered for executing CHI-BD were the default parameters available in the original implementation of the algorithm[2]. It was considered three linguistic

[1]*http : //sites.labic.icmc.usp.br/text_collections/*

[2]https://github.com/melkano/CHI-BD

Table 5.1 Summary of the datasets.

Dataset	#Examples	(#maj;#min)	#Features
20ng_0	1000	(500;500)	8526
20ng_1	2000	(1500;500)	11027
20ng_2	2000	(1500;500)	11027
20ng_3	2000	(1500;500)	11027
20ng_4	2000	(1500;500)	11027
Ohscal	2881	(1621;1260)	8214
Re8	6215	(3923;2292)	7846
Multidomain sentiment	8000	(4000;4000)	13360

labels per attribute, and it was applied the winning rule fuzzy reasoning method for classifying examples. Rule weights have been computed using the *Penalized Cost-Sensitive Certainty Factor* (PCF-CS) (LÓPEZ et al., 2015), which is an adaptation of the *Penalized Certainty Factor* (PFC) (ISHIBUCHI; YAMAMOTO, 2005). To conduct the experiments it was used an Ubuntu virtual machine Version 16.04.1 with 256 GB of RAM memory and 32 cores, operating on a VirtualBox 5.2.1 platform under a machine equipped with an Intel(R) Xeon(R) CPU E7-2890 v2 processor at 2.80GHz.

The performance of the *Summarizer* technique will be assessed by means of geometric means (Gmeans) (GALAR et al., 2011), a well known metric for imbalanced datasets, and also by means of the number of rules.

FCM was used in the horizontal dimensionality reduction step to create groups of attributes varying from 2 and 10 groups to proportions of 1%, 5% and 10% of the number of attributes. The groups of attributes were then used as new attributes for the CHI-BD algorithm, which configures the *Summarizer* approach. The performance of CHI-BD with the reuced datasets with different number of groups was then compared to the traditional approach (CHI-BD with no horizontal dimensionality reduction). The horizontal dimensionality reduction step was also performed using PCA and LSA under the same conditions.

Comparisons in terms of number of rules generated by CHI-BD for each dataset and for each method are discussed in the next section.

5.2 NUMBER OF RULES

Figure 5.1 shows the number of rules generated by CHI-BD without *Summarizer* for each of the 9 datasets considered in this work, as well as the number of rules generated by CHI-BD for different variations of *Summarizer* (2 and 10 groups, generated by FCM, PCA and LSA, to proportions of 1%, 5% and 10% of the number of attributes).

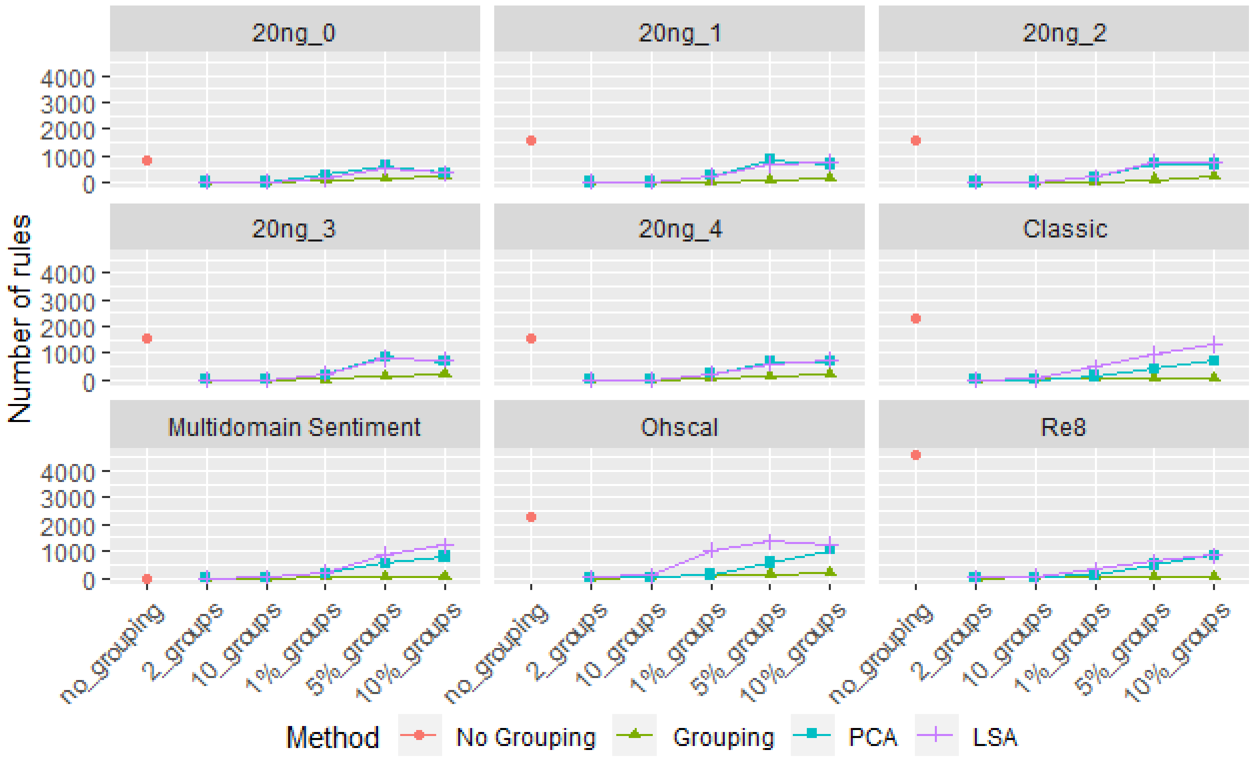

Figure 5.1 Number of rules generated by the CHI-BD algorithm.

According to the results in Figure 5.1, the number of rules generated by CHI-BD with no grouping step (without *Summarizer*) was higher for all datasets when compared to *Summarizer* with FCM, PCA and LSA. The number of rules for CHI-BD with no grouping step for the Multidomain Sentiment dataset could not be counted because the number of rules was to high to be written in the Rule Base file. Among the *Summarizer* approaches used with CHI-BD, FCM returned the smaller number of rules when compared to PCA and LSA for all datasets.

It can be observed that, for same datasets, the number of rules generated by CHI-BD decreases for *Summarizer* with 10% of the number of attributes. That can be an indication that redundancy starts being added to the model when a high number of groups of terms is considered. This way, a smaller number of groups may carry all the information about the data and adding more groups only increases dimensionality with no extra information gain.

The average number of rules for every number of groups generated by *Summarizer* is presented in Table 5.2. The average number of rules generated by CHI-BD without *Summarizer* was equal to 1813. The percentage of decrease in the number of rules obtained by CHI-BD with *Summarizer* compared to CHI-BD without *Summarizer* (1813) is also presented in table 5.2 (percentages in parenthesis). In order to find statistical differences between the number of rules, the Wilcoxon test(WILCOXON, 1992) was applied. The p-values are also presented in Table 5.2.

As can be seen in Table 5.2, among FCM, PCA and LSA, *Summarizer* with FCM presented the smaller average number of rules. For *Summarizer* with 10% of the number of terms, for example, the average number of rules for FCM was equal to 147 rules, while the average number of rules was equal to 722 and 888 for *Summarizer* with PCA and LSA, respectively. That is, with the same amount of attributes (groups), CHI-BD with *Summarizer* considering FCM was capable of designing a classification system with a much smaller number of rules. The actual number of rules for every tested approach is presented in Table 7.1 in the Appendix.

Table 5.2 P-values, average number of rules for CHI-BD with *Summarizer* and its percentage of decrease when compared to CHI-BD without *Summarizer*

Summarizer		**Average Number of Rules**		
Number of groups	**p-value**	*FCM*	*PCA*	*LSA*
Summarizer 2	< 0.0001	3 (99.8%)	4 (99.8%)	5 (99.7%)
Summarizer 10	< 0.0001	10 (99.4%)	15 (99.2%)	28 (98.4%)
Summarizer 1%	< 0.0001	48 (97.4%)	192 (89.4%)	349 (80.8%)
Summarizer 5%	< 0.0001	108 (94.0%)	636 (64.9%)	804 (55.6%)
Summarizer 10%	< 0.0001	147 (92.0%)	722 (60.2%)	888(51.0%)

Comparisons of the methods in terms of Gmeans (GALAR et al., 2011) will be presented in next section, and one should keep in mind that there should be a balance between classification accuracy and rule base size, since the number of rules in a classification system play an important role in the interpretability of the results.

The number of rules for CHI-BD with FCM, PCA and LSA drops at a high rate for all approaches of *Summarizer* when compared to CHI-BD without *Summarizer*. The percentage of decrease on the number of rules of *Summarizer* with FCM was above 90% for all number of groups of terms. Based on the results, it can be observed a significant reduction on the number of rules of CHI-BD with *Summarizer* when compared to CHI-BD without *Summarizer* with a significance level of 99% (p-value < 0.0001) in all cases. That is an expected behavior since a smaller number of attributes are being used for constructing the Rule Base.

Despite the significant decrease on the number of rules for CHI-BD with *Summarizer*, there was not a negative impact on the performance of the algorithm, as discussed in the next section.

5.3 CLASSIFICATION PERFORMANCE

Table 5.3 shows the Gmeans values obtained by CHI-BD with and without *Summarizer* for each of the 9 datasets considered in this work. The best overall result for each dataset is shown underlined, while the best one for each dataset and for each number of groups on *Summarizer* among FCM, PCA and LSA is shown in bold-face.

As shown in Table 5.3, for 8 out of the 9 studied datasets, the performance of CHI-BD with *Summarizer* was better than the performance of CHI-BD without *Summarizer* in terms of Gmeans for at least one of the horizontal dimensionality reduction approaches (FCM, PCA or LSA). For most of the cases, CHI-BD with *Summarizer* with only 2 groups of attributes already results in better Gmeans than CHI-BD without *Summarizer*.

For most of the datasets, it can be found a percentage of reduction on the number of terms that results in an increase on the Gmeans metric for some of the FCM, PCA and LSA method in comparison to CHI-BD without *Summarizer*. It should be noted that there was a significant reduction in the number of rules for CHI-BD with *Summarizer* in addition to the increase of Gmeans. The best possible method would be the one with the higher Gmeans and the smaller number of rules.

For the dataset Ohscal, for example, CHI-BD with no horizontal reduction resulted in a Gmeans value of 0.7501 with a Rule Base composed of 2305 rules. Each rule for CHI-BD with no horizontal reduction applied to the Ohscal dataset was composed of 8214 linguistic variables (one for each feature of the dataset). On the other hand, for the same dataset, CHI-BD with LSA with only 10 groups (semantic concepts) resulted in a Gmeans value of 0.8147. That is a Gmeans value 8.6% higher for a much smaller number of linguistic variables for each rule (10 linguistic variables for each rule, in this case). The number of rules in the reduced scenario was also very smaller, as expected (84 rules). In summary, applying *Summarizer* with LSA and 10 groups resulted in an increase of 8.6% on the Gmeans and a decrease of 96.3% on the number of rules when comparing CHI-BD with no horizontal dimensionality reduction.

In some other cases, however, reducing horizontal high dimensionality did not result in better Gmeans. Taking the same dataset, Ohscal, for example, CHI-BD with LSA and 5% of groups resulted in a Gmeans value of 0.7317, while CHI-BD with no horizontal reduction resulted in a Gmeans value of 0.7501. That represents a reduction of 2.4% on

Table 5.3 Gmeans for CHI-BD with and without *Summarizer*.

		Gmeans		
Dataset	**Method**	*FCM*	*PCA*	*LSA*
20ng$_0$	No horizontal reduction		.7071	
	Summarizer 2	.7051	.7195	**.7232**
	Summarizer 10	**.7121**	.7029	.7089
	Summarizer 1%	.6071	**.7367**	.7361
	Summarizer 5%	.5689	.6570	**.7293**
	Summarizer 10%	.5881	.7060	**.7071**
20ng$_1$	No horizontal reduction		**.7500**	
	Summarizer 2	.4755	.4917	.4654
	Summarizer 10	.5006	.5005	.5005
	Summarizer 1%	.3567	.3198	.5466
	Summarizer 5%	.4779	.1885	.4018
	Summarizer 10%	.4890	.1000	-
20ng$_2$	No horizontal reduction		-	
	Summarizer 2	**.5004**	.4620	.5064
	Summarizer 10	**.5018**	.5015	.5015
	Summarizer 1%	.3007	.5312	**.5336**
	Summarizer 5%	.2718	**.2946**	.2604
	Summarizer 10%	**.2575**	-	-
20ng$_3$	No horizontal reduction		0.1000	
	Summarizer 2	.1684	**.3484**	.1912
	Summarizer 10	.1013	**.2940**	.1000
	Summarizer 1%	.2901	.5071	**.5203**
	Summarizer 5%	.2961	.3028	**.3229**
	Summarizer 10%	**.3612**	-	.0000
20ng$_4$	No horizontal reduction		.1540	
	Summarizer 2	.1142	.2925	**.4106**
	Summarizer 10	**.4778**	-	.1278
	Summarizer 1%	**.4916**	.3324	.3342
	Summarizer 5%	**.5082**	.3590	.3924
	Summarizer 10%	**.4926**	.1414	-
Classic$_0$	No horizontal reduction		.7147	
	Summarizer 2	.6802	**.9751**	.8888
	Summarizer 10	.7073	**.9670**	.9173
	Summarizer 1%	.7080	**.8278**	.8076
	Summarizer 5%	**.7343**	.2528	.7300
	Summarizer 10%	**.7277**	.4726	.7163
Ohscal	No horizontal reduction		.7501	
	Summarizer 2	.5549	.6560	**.8124**
	Summarizer 10	.6707	.3621	**.8147**
	Summarizer 1%	.7712	.3703	**.7934**
	Summarizer 5%	.6795	.6787	.7317
	Summarizer 10%	.7246	.7497	.7488
Re8	No horizontal reduction		.8034	
	Summarizer 2	.3696	.7945	.7979
	Summarizer 10	.2774	**.8296**	.8013
	Summarizer 1%	.2726	.7910	**.8077**
	Summarizer 5%	.4337	.7989	.8022
	Summarizer 10%	.7199	.7942	.7949
Multi-domain sentiment	No grouping		.7071	
	Summarizer 2	.6825	**.7072**	.3850
	Summarizer 10	.3136	**.7072**	.6968
	Summarizer 1%	.0641	.7060	.7035
	Summarizer 5%	.2620	.2437	.3502
	Summarizer 10%	.5792	.7071	**.7075**

the Gmeans value when using CHI-BD with *Summarizer*. However, not only the Gmeans value should be taken into account when deciding wich approach to choose. In this case, despite the decrease of the Gmeans value, there was a gain with the smaller Rule Base and smaller size of each rule. For this example, there was a reduction of 40.95% on the number of rules for CHI-BD without *Summarizer* (2305 rules) versus CHI-BD with *Sumarizer* (1361 rules).

Table 5.4 presents a comparison between the number of times CHI-BD with *Summarizer* had a better Gmeans performance than CHI-BD without *Summarizer*. The results are presented in terms of wins (W), which is the number of times the use of some reduction approach obtained a classification result superior to the CHI-BD without *Summarizer*; ties (T), which is the number of times the use of some reduction approach obtained a classification result similar to the CHI-BD without *Summarizer*; and loses (L), which is the number of times the use of some reduction approach obtained a classification result inferior to the CHI-BD without *Summarizer*. Finally, the Total row indicates the total number of wins, ties and loses considering all possible number of groups for *Summarizer*.

A total column was added to Table 5.4 in order to consider the number of wins, ties and loses when at least of the three approaches presents a better Gmeans than CHI-BD without *Summarizer*. For example, a win is considered in the Total column when FCM or PCA or LSA presented a better Gmeans than CHI-BD without *Summarizer*.

Table 5.4 Gmeans for CHI-BD with and without *Summarizer*

Summarizer	**Gmeans W/T/L**			
Number of groups	Total	*FCM*	*PCA*	*LSA*
Summarizer 2	7/0/2	3/0/6	7/0/2	7/0/2
Summarizer 10	8/0/1	4/0/5	5/0/4	4/1/4
Summarizer 1%	7/0/2	4/0/5	5/0/4	7/0/2
Summarizer 5%	5/0/4	4/0/5	3/0/6	5/0/4
Summarizer 10%	5/1/3	4/0/5	1/1/7	3/1/5
TOTAL	32/1/12	19/0/26	23/1/21	26/2/17

As shown in Table 5.4, despite the much smaller number of rules obtained by CHI-BD with *Summarizer*, it still gives a higher Gmeans value in 32 out of 45 cases (71.1% of wins). LSA presented the higher number of wins (26 wins). For the Total column, *Summarizer* with 10 groups presented the higher number of wins (8 out of 9 datasets), followed by *Summarizer* 2 and 1% with 7 wins. As shown in Table 5.4, it was possible to obtain a higher Gmeans value when using *Summarizer* for most of the cases when considering at least one of the approaches. One should keep in mind that there should be a balance between model performance and model complexity. This way, choosing the model with the higher Gmeans value is not always the best choice.

To summarize, in this Chapter we compared the results of the classification process with and without horizontal reduction (with and without *Summarizer*) in terms of number of rules and classification performance. *Summarizer* was proven to significantly reduce the number of rules generated by CHI-BD, since a much smaller feature space is considered instead of the original set of attributes. Despite the significant reduction on the number of features, and consequently, on the number of rules generated by the model, there was not a negative impact on the performance of the algorithm in terms of Gmeans.

In the next Chapter, it will be presented the conclusions of this work.

Chapter 6

CONCLUSIONS

A huge amount of data is being generated everyday with the advent of technology. These data, often called Big Data, come in a variety of formats and sizes and are usually not easy for dealing with. One of the challenges that arises with Big Data is its high dimensionality. The vertical and horizontal high dimensionality that can come with Big Data prevent traditional ML algorithms to be able to extract useful information from such data. Two of the reasons that makes it harder for traditional ML to be able to deal with Big Data are processing time and memory consumption. Such algorithms, as Chi et al.'s algorithm, were not designed to process huge amounts of data in a reasonable amount of time and with reasonable memory consumption.

For having the benefits of using fuzzy rule based classifiers on Big Data as well, researchers have tried to design fuzzy classifiers that reduce memory and computational requirements on Big Data classification tasks, as exposed by Elkano et al. (2019) (Elkano; Bustince; Galar, 2019). Most of the classifiers designed for Big Data make use of the MapReduce paradigm and the Hadoop computing framework. MapReduce model works by dividing data into blocks and processing each block of data independently and at the same time, while Hadoop implements the MapReduce model over a distributed file system. This approach makes it possible to process huge amounts of data in a reasonable period of time. However, recent proposed algorithms often consider Big Data as datasets with large number of instances, with no concern about the high number of attributes as well.

Text datasets, for example, can have vertical and horizontal high dimensionality at the same time. This way, recent adaptations of fuzzy rule based systems are not appropriate for dealing with such datasets, since only vertical high dimensionality is tackled by these recent proposals. All adapted algorihtms for Big Data classification problems discussed in this work were only tested on datasets with a relatively small feature space. The largest number of attributes tested by these recent adaptations had only 54 attributes. For text datasets, since every possible word in all documents may became an attribute, the feature space can get much higher than 54 attributes. One way to reduce horizontal high dimensionality is by using feature selection techniques to choose a reduced number of attributes to compose the feature space. However, the existing feature selection processes might not guarantee a fair representation of all classes by the selected features, specially for imbalanced datasets. These techniques evaluate each attribute individually, and dependencies between attributes might be ignored, resulting in the selection of redundant attributes and on the discard of relevant ones (CHANDRASHEKAR; SAHIN, 2014; KUMBHAR; MALI, 2016). In addition to that, even if a smaller number of attributes is selected, it might not be small enough for such algorithms.

With the above mentioned points, it is clear that there is a lack of approaches for dealing with both horizontal and vertical high dimensional datasets. In this sense, the following hypothesis guided the developement of this work:

It is possible to reduce dimensionality, without discarding information, and build better models by using groups of attributes in the classification process in a fuzzy rule based classification system using the MapReduce paradigm

Therefore, we have developed a new approach for building reduced feature spaces of vertical and horizontal high dimensional datasets. We named our approach as *Summarizer*, and we have presented details of its procedure in Chapter 5.

To assess the feasibility of the hypothesis of this work, we tested our approach with CHI-BD, that already deals with vertical high dimensionality by making use of the MapReduce paradigm. Three different approaches of *Summarizer* (FCM, PCA and LSA) were tested in different datasets of different sources. The results were compared to the classification process with no horizontal reduction (without *Summarizer*) in terms of number of rules and classification performance.

As a result, our method was proven to significantly reduce the number of rules generated by CHI-BD, since a much smaller feature space is considered instead of the original set of attributes. Despite the significant reduction on the number of features, and consequently, on the number of rules generated by the model, there was not a negative impact on the performance of the algorithm in terms of Gmeans. Therefore, it is considered that the hypothesis of this work was confirmed.

This chapter presents the final considerations of this work. Therefore, the scientific contributions, limitations and further research will be discussed in the next sections.

6.1 CONTRIBUTIONS

The contribution of this work to the state of the art classification algorithms for Big Data includes a new approach for reducing horizontal high dimensionality.

The main scientific contribution of this work consists of the idealization, development, and assessment of *Summarizer*, an approach for reducing horizontal dimensionality by defining groups of attributes as the new feature space. *Summarizer* was built with the goal of enabling the use of existing ML algorithms that already deal with the vertical high dimensionality of Big Data sets. The results show that the proposed approach is effective for reducing horizontal high dimensionality without discarding information, as well as for creating smaller rule-based models without impacting on the performance of the algorithms.

Also, the manuscript of this work has been submitted to FUZZ-IEEE and is at the revision stage.

6.2 LIMITATIONS AND FURTHER RESEARCH

The first limitation of this work is the process of choosing the appropriate number of groups for representing the new feature space. The definition of the appropriate parameters in a non supervised approach is usually done empirically. However, when dealing with Big Data, empirically defining the proper number of groups may be very computational costly. This way, a suggestion for future work is to design a better approach for defining the parameters of the model.

Another limitation of this work is in the process of building groups of attributes. It was proposed the use of the FCM algorithm which is very computational costly. Therefore, another direction for future work is considering a MapReduce approach when building the reduced feature space.

Another suggestion for future work is to test the *Summarizer* approach with algorithms other than CHI-BD. FRBCSs were the focus of this work, however, other classification algorithms that deal only with vertical high dimensionality might take advantage of *Summarizer* when dealing with horizontal and vertical high dimensional datasets. In addition to that, *Summarizer* should also be considered for reducing horizontal high dimensionality before applying traditional algorithms that does not deal with Big Data, since *Summarizer* is an approach for reducing horizontal high dimensionality and can be applied to any classification algorithm.

BIBLIOGRAPHY

ANCHALIA, P. P.; ROY, K. The k-nearest neighbor algorithm using mapreduce paradigm. In: IEEE. *Intelligent Systems, Modelling and Simulation (ISMS), 2014 5th International Conference on*. [S.l.], 2014. p. 513–518.

BEZDEK, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms*. USA: Kluwer Academic Publishers, 1981. ISBN 0306406713.

CHANDRASHEKAR, G.; SAHIN, F. A survey on feature selection methods. *Computers & Electrical Engineering*, Elsevier, v. 40, n. 1, p. 16–28, 2014.

CHEN, C. et al. 6 million spam tweets: A large ground truth for timely twitter spam detection. In: *IEEE International Conference on Communications (ICC)*. London, UK: IEEE, 2015. p. 7065–7070.

CHI, Z.; YAN, H.; PHAM, T. *Fuzzy algorithms: with applications to image processing and pattern recognition*. [S.l.]: World Scientific, 1996.

CHU, C.-T. et al. Map-reduce for machine learning on multicore. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2007. p. 281–288.

DAI, W.; JI, W. A mapreduce implementation of c4. 5 decision tree algorithm. *International journal of database theory and application*, v. 7, n. 1, p. 49–60, 2014.

DEAN, J.; GHEMAWAT, S. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, ACM, v. 51, n. 1, p. 107–113, 2008.

DENG, X. et al. Feature selection for text classification: A review. *Multimedia Tools and Applications*, Springer, p. 1–20, 2018.

DUBOIS, D.; PRADE, H. What are fuzzy rules and how to use them. *Fuzzy sets and systems*, Elsevier, v. 84, n. 2, p. 169–185, 1996.

Elkano, M.; Bustince, H.; Galar, M. Do we still need fuzzy classifiers for small data in the era of big data? In: *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. [S.l.: s.n.], 2019. p. 1–6. ISSN 1544-5615.

ELKANO, M. et al. Chi-bd: A fuzzy rule-based classification system for big data classification problems. *Fuzzy Sets and Systems*, Elsevier, 2017.

ELKANO, M. et al. A global distributed approach to the chi et al. fuzzy rule-based classification system for big data classification problems. In: IEEE. *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*. [S.l.], 2017. p. 1–6.

FAHAD, A. et al. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing*, IEEE, v. 2, n. 3, p. 267–279, 2014.

FERNÁNDEZ, A. et al. Fuzzy rule based classification systems for big data with mapreduce: granularity analysis. *Advances in Data Analysis and Classification*, Springer, v. 11, n. 4, p. 711–730, 2017.

FERNÁNDEZ, A. et al. Big data with cloud computing: an insight on the computing environment, mapreduce, and programming frameworks. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 4, n. 5, p. 380–409, 2014.

GALAR, M. et al. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, IEEE, v. 42, n. 4, p. 463–484, 2011.

GARCÍA, S.; LUENGO, J.; HERRERA, F. *Data preprocessing in data mining.* [S.l.]: Springer, 2015.

GOLUB, G. H.; REINSCH, C. Singular value decomposition and least squares solutions. In: *Linear Algebra.* [S.l.]: Springer, 1971. p. 134–151.

HAN, J.; LIU, Y.; SUN, X. A scalable random forest algorithm based on mapreduce. In: IEEE. *Software Engineering and Service Science (ICSESS), 2013 4th IEEE International Conference on.* [S.l.], 2013. p. 849–852.

ISHIBUCHI, H.; NAKASHIMA, T.; NII, M. *Classification and modeling with linguistic information granules: Advanced approaches to linguistic Data Mining.* [S.l.]: Springer Science & Business Media, 2006.

ISHIBUCHI, H.; YAMAMOTO, T. Rule weight specification in fuzzy rule-based classification systems. *IEEE transactions on fuzzy systems*, IEEE, v. 13, n. 4, p. 428–435, 2005.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, The Royal Society Publishing, v. 374, n. 2065, p. 20150202, 2016.

KHAIRNAR, J.; KINIKAR, M. Sentiment analysis based mining and summarizing using svm-mapreduce. *International Journal of Computer Science and Network Security (IJCSNS)*, International Journal of Computer Science and Network Security, v. 15, n. 4, p. 90, 2015.

KIRAN, M. et al. Verification and validation of mapreduce program model for parallel support vector machine algorithm on hadoop cluster. *International Journal of Computer Science Issues*, Citeseer, v. 10, n. 1, p. 317–325, 2013.

KLIR, G.; YUAN, B. *Fuzzy sets and fuzzy logic.* [S.l.]: Prentice hall New Jersey, 1995.

KUMBHAR, P.; MALI, M. A survey on feature selection techniques and classification algorithms for efficient text classification. *International Journal of Science and Research*, v. 5, n. 5, p. 9, 2016.

LABANI, M. et al. A novel multivariate filter method for feature selection in text classification problems. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 70, p. 25–37, 2018.

LANDAUER, T. K.; FOLTZ, P. W.; LAHAM, D. An introduction to latent semantic analysis. *Discourse processes*, Taylor & Francis, v. 25, n. 2-3, p. 259–284, 1998.

LI, B. et al. Scalable random forests for massive data. In: SPRINGER. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. [S.l.], 2012. p. 135–146.

LI, J. et al. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, ACM, v. 50, n. 6, p. 94, 2017.

LOPEZ, V. et al. On the use of mapreduce to build linguistic fuzzy rule based classification systems for big data. In: IEEE. *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*. [S.l.], 2014. p. 1905–1912.

LÓPEZ, V. et al. Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data. *Fuzzy Sets and Systems*, Elsevier, v. 258, p. 5–38, 2015.

MAILLO, J. et al. knn-is: An iterative spark-based design of the k-nearest neighbors classifier for big data. *Knowledge-Based Systems*, Elsevier, v. 117, p. 3–15, 2017.

MAILLO, J.; TRIGUERO, I.; HERRERA, F. A mapreduce-based k-nearest neighbor approach for big data classification. In: IEEE. *Trustcom/BigDataSE/ISPA, 2015 IEEE*. [S.l.], 2015. v. 2, p. 167–172.

MAILLO, J.; TRIGUERO, I.; HERRERA, F. A mapreduce-based k-nearest neighbor approach for big data classification. In: IEEE. *Trustcom/BigDataSE/ISPA, 2015 IEEE*. [S.l.], 2015. v. 2, p. 167–172.

MAMDANI, E. H. Application of fuzzy logic to approximate reasoning using linguistic synthesis. In: IEEE COMPUTER SOCIETY PRESS. *Proceedings of the sixth international symposium on Multiple-valued logic*. [S.l.], 1976. p. 196–202.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, v. 5, n. 4, p. 1093 – 1113, 2014.

NAKASHIMA, T. et al. A weighted fuzzy classifier and its application to image processing tasks. *Fuzzy sets and systems*, Elsevier, v. 158, n. 3, p. 284–294, 2007.

NICOLETTI, M. d. C.; CAMARGO, H. d. A. Fundamentos da teoria de conjuntos fuzzy. *São Carlos: EdUFSCar*, 2004.

PEARSON, K. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 2, n. 11, p. 559–572, 1901.

PEDRYCZ, W.; GOMIDE, F. *An introduction to fuzzy sets: analysis and design.* [S.l.]: Mit Press, 1998.

PRIYADARSHINI, A. et al. A map reduce based support vector machine for big data classification. *International Journal of Database Theory and Application*, v. 8, n. 5, p. 77–98, 2015.

PRIYADARSHINI, A. et al. A map reduce based support vector machine for big data classification. *International Journal of Database Theory and Application*, v. 8, n. 5, p. 77–98, 2015.

RÍO, S. D. et al. On the use of mapreduce for imbalanced big data using random forest. *Information Sciences*, Elsevier, v. 285, p. 112–137, 2014.

RIO, S. del et al. A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. *International Journal of Computational Intelligence Systems*, Taylor & Francis, v. 8, n. 3, p. 422–437, 2015.

SANZ, J. A. et al. A compact evolutionary interval-valued fuzzy rule-based classification system for the modeling and prediction of real-world financial applications with imbalanced data. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 23, n. 4, p. 973–990, 2015.

SANZ, J. A. et al. Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. *Applied Soft Computing*, Elsevier, v. 20, p. 103–111, 2014.

SARAÇ, E.; ÖZEL, S. A. An ant colony optimization based feature selection for web page classification. *The Scientific World Journal*, Hindawi, v. 2014, 2014.

SAXENA, A. et al. A review of clustering techniques and developments. *Neurocomputing*, Elsevier, v. 267, p. 664–681, 2017.

SEGATORI, A.; MARCELLONI, F.; PEDRYCZ, W. On distributed fuzzy decision trees for big data. *IEEE Transactions on Fuzzy Systems*, IEEE, v. 26, n. 1, p. 174–192, 2018.

SILVA, J. M. B.; SILVA, F. Feature extraction for the author name disambiguation problem in a bibliographic database. In: *Proceedings of the Symposium on Applied Computing.* New York, NY, USA: ACM, 2017. (SAC '17), p. 783–789.

STREHL, A.; GHOSH, J.; MOONEY, R. Impact of similarity measures on web-page clustering. In: *Workshop on artificial intelligence for web search (AAAI 2000).* [S.l.: s.n.], 2000. v. 58, p. 64.

SUN, Z.; FOX, G. Study on parallel svm based on mapreduce. In: THE STEERING COMMITTEE OF THE WORLD CONGRESS IN COMPUTER SCIENCE, COMPUTER ENGINEERING AND APPLIED COMPUTING (WORLDCOMP). *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA).* [S.l.], 2012. p. 1.

TAKAGI, T.; SUGENO, M. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man, and cybernetics*, IEEE, n. 1, p. 116–132, 1985.

WAKAYAMA, R. et al. Distributed forests for mapreduce-based machine learning. In: IEEE. *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on.* [S.l.], 2015. p. 276–280.

WANG, R. et al. Learning elm-tree from big data based on uncertainty reduction. *Fuzzy Sets and Systems*, Elsevier, v. 258, p. 79–100, 2015.

WHITE, T. *Hadoop: The definitive guide.* [S.l.]: " O'Reilly Media, Inc.", 2012.

WILCOXON, F. Individual comparisons by ranking methods. In: *Breakthroughs in statistics.* [S.l.]: Springer, 1992. p. 196–202.

ZADEH, L. A. Information and control. *Fuzzy sets*, v. 8, n. 3, p. 338–353, 1965.

ZHANG, C.; LI, F.; JESTES, J. Efficient parallel knn joins for large data in mapreduce. In: ACM. *Proceedings of the 15th International Conference on Extending Database Technology.* [S.l.], 2012. p. 38–49.

CHAPTER 7

APPENDIX

Table 7.1 shows the number of rules generated by CHI-BD for each dataset with all the different approaches of *Summarizer* as well as with CHI-BD with no horizontal dimensionality reduction.

Table 7.1 Number of rules generated by CHI-BD for each of the tested approaches and for each dataset.

Dataset	**Method**	**# Rules FCM**	**# Rules PCA**	**# Rules LSA**
20ng$_0$	No horizontal reduction		795	
	Summarizer 2	3	5	4
	Summarizer 10	3	11	10
	Summarizer 1%	60	276	152
	Summarizer 5%	176	621	566
	Summarizer 10%	201	367	362
20ng$_1$	No horizontal reduction		1584	
	Summarizer 2	3	4	3
	Summarizer 10	3	13	10
	Summarizer 1%	30	247	226
	Summarizer 5%	114	852	670
	Summarizer 10%	178	685	759
20ng$_2$	No horizontal reduction		1584	
	Summarizer 2	3	4	4
	Summarizer 10	4	13	9
	Summarizer 1%	24	209	208
	Summarizer 5%	120	714	782
	Summarizer 10%	204	681	758
20ng$_3$	No horizontal reduction		1584	
	Summarizer 2	3	3	3
	Summarizer 10	3	15	11
	Summarizer 1%	27	202	241
	Summarizer 5%	118	853	808
	Summarizer 10%	204	682	758
20ng$_4$	No horizontal reduction		1584	
	Summarizer 2	3	4	4
	Summarizer 10	4	15	13
	Summarizer 1%	29	247	244
	Summarizer 5%	111	684	611
	Summarizer 10%	187	683	758
Classic	No horizontal reduction		2283	
	Summarizer 2	3	4	7
	Summarizer 10	31	17	68
	Summarizer 1%	60	144	486
	Summarizer 5%	55	404	945
	Summarizer 10%	67	715	1323
Ohscal	No horizontal reduction		2305	
	Summarizer 2	3	4	7
	Summarizer 10	26	18	84
	Summarizer 1%	131	120	1037
	Summarizer 5%	152	567	1361
	Summarizer 10%	158	1043	1235
Re8	No horizontal reduction		4594	
	Summarizer 2	3	4	6
	Summarizer 10	10	18	32
	Summarizer 1%	33	105	322
	Summarizer 5%	69	495	621
	Summarizer 10%	63	846	838
Multi-domain Sentiment	No horizontal reduction		-	
	Summarizer 2	3	3	3
	Summarizer 10	3	16	18
	Summarizer 1%	35	175	222
	Summarizer 5%	59	533	869
	Summarizer 10%	59	793	1200