Received: August 3, 2019

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Novel Data Mining Approach for Big Data Based on Rough Sets and Fuzzy Logic

Osama Sayed Abdelrahman[1]* **Hesham Ahmed Hefny[1]**

[1]*Faculty of Graduate Studies for Statistical Research, Cairo University, Egypt*
* Corresponding author's Email: osayed72@yahoo.com

Abstract: The term "Big Data" is a buzzword which describes new technologies that manipulate very large data sets which are massively generated by heterogonous sources. This new term encourages data scientists to extend their work and modify their techniques to overcome the new challenges come with big data concepts. Granular computing has emerged as a new rapidly growing information processing paradigm inside the community of Computational Intelligence. Theories of Fuzzy sets and Rough sets theory are considered powerful examples of granular computing that can be applied to data mining techniques to extract nontrivial knowledge from huge data. The aim of this paper is to introduce a data mining approach for big data based on integrating fuzzy sets and rough sets theories. The proposed approach provides a novel granular data mining for big data that allow extracting useful knowledge and rules from huge data to enhance the decision making process. The proposed approach has been applied on different types of datasets. The experimental results show that our proposed approach is more efficient and robust when dealing with very big datasets and obtained consistent classification rules with classification accuracy 100%.

Keywords: Big data, Data mining, Granular computing, Rough sets, Fuzzy sets, Hadoop, Spark, Spark SQL.

1. Introduction

Nowadays, with the very large amount of generated data from different sources and devices, the process of extracting valuable information and nontrivial knowledge from such huge sizes of data becomes one of the important and interesting research directions. Recently, big data analysis and mining is attracting more attention for both scientific and industrial areas. In the last decade, Big Data technologies appeared as distributed computing technologies that allow processing large amount of data on clusters of commodity hardware. Such a distributed and parallelized processing led to high scalability, high efficiency, fault-tolerance and load balancing of huge size data sets. Google Firstly introduced MapReduce as a programming model and Apache Hadoop from Yahoo is considered the first implementation of MapReduce that becomes the most popular platform of large scale distributed data processing [1].

Granular computing has emerged as a new rapidly growing information processing paradigm under the umbrella of Computational Intelligence. The basic idea of granular computing is that subsets of a universe of discourse, or equivalent classes or a modules of a given information system can be viewed as granules. Granules can be considered as interconnected information units through which the whole universe can be handled as a whole. Such granules can also be decomposed into smaller granules called sub-granules. This allows discovering various forms of rational decisions at different levels of parameter quantization and system variable resolutions. Such a utilization of tolerance of imprecision leads to powerful manipulation of uncertainty in real world decision making problems. Fuzzy sets theory and Rough sets theory represent two famous approaches of granular computing that are successfully applied to uncertain decision making problems as well as data mining applications [2-5].

The main objective of this paper is to introduce a novel approach for mining big data based on integrating big data technologies with granular computing, in particular fuzzy sets and rough sets theories to generate fuzzy decision rules from big data.

Such an approach helps in extracting hidden knowledge in big data. The rest of this paper is organized as follows. Section 2 introduces the basic background of Granular Computing and big data technologies. In section 3, we introduce related work. In section 4, we propose the basic idea of integration between granular computing and data mining. Section 5 presents the evaluation of the proposed approach. Finally, the paper ends with conclusion in section 6.

2. Background

In this section, the basic concepts of granular computing and big data technologies are introduced.

2.1 Granular computing

Granular computing can be defined as a general problem solving theory based on various levels of granularity and details. The term "Granular Computing" is firstly introduced in 1997, but the main ideas of granular computing have been studied in many research fields with different names. Rough sets theory, fuzzy sets theory, data analysis, data mining, and machine learning are the most famous names of these fields which use granular computing principals [3].

Zadeh firstly introduced the notation of information granulation in 1979 and he suggested that his theory of fuzzy sets can find potential application in this respect. In 1982, Pawlak introduced rough sets theory which provides solid example of granular computing [3]. In 1997, Zadeh recalled information granulation to renew the researcher's interest [4]. Lin was the first who coined the buzzword "Granular Computing" for this newly emergent research field in 1997 [5].

As a base for data mining, granular computing can help for extracting hidden knowledge. Rules are one of the most useful representations of knowledge discovery from huge data. One of the most important key notations of fuzzy sets theory is linguistic variables which are considered fuzzy granules and can be represented by natural language. By using fuzzy granules, i.e. "linguistic variables", the extracted rules become more human friendly, more understandable and more useful since they provide a smooth transition between member and non-member of a set which is easily understandable to human [4]. In order to mine more meaningful rules we can group attribute values into granules. On the other hand, granular computing can be combined with various data mining methods to get more effective and efficient methods for extracting more useful rules from huge size of data sets.

Rough sets theory is also applied to data mining as a strong example of granular computing. Rough sets are considered as one of the powerful data analysis techniques and applied successfully in knowledge discovery and data mining [5]. We will summarize the basic notation of rough sets as follows [6]:

Definition 1 (*Information System*): Information system is the basic notation of rough set data analysis in which the data set is represented as a table, where each row represents an object. Every column represents an attribute that can be measured for each object. Information system $S = [U, A, V, f]$, where U is the universe; A is a finite set of attributes, V the domain of attribute a, f is an information function $U \times A \to V$.

Definition 2 (*Decision Table*): An Information System $S = [U, A, V, f]$ is called Decision table if A is the union set of condition attributes C and decision attribute D.

Definition 3 (*Equivalence Relation*): An approximation space induces a granulated view of the universe, the equivalence relation E on U, $E \subseteq U \times U$.

Definition 4 (*Equivalence Class*): Under an equivalence relation, equivalence classes are the smallest non-empty definable subsets of U. For an object $x \in U$, the equivalence class containing x is given by:

$$[x]_E = \{ y \mid xEy \} \tag{1}$$

Definition 5 (*lower and upper approximations*): Using $[x]_E$ the equivalence class containing x we can define the lower and upper approximations of X by E in A as:

$$\underline{A}(X) = \{ x \in U : [x]_E \subset X \} \tag{2}$$

$$\overline{A}(X) = \{ x \in U : [x]_E \cap X \neq \emptyset \} \tag{3}$$

The objects in $\underline{A}(x)$ can be with certainty classified as members of on the basis of knowledge in$R$, while the objects in $\overline{A}(x)$can be only classified as possible members of Xon the basis of knowledge in R.

Definition 6 (*Reducts and Core*): A reduct is subset of condition attributes that can discern all

objects. The reduct set is the minimal subset of condition attributes that has the same classification power as the entire condition attributes. The core is the intersection of all reducts.

2.2 Big data tools and technologies

In 2004, Google firstly introduced the MapReduce model as a high-level programming model for distributed computing model that deals with large scale parallel data processing [7]. This model introduces two main functions *Map* and *Reduce*, the first function *Map* takes as input a set of key-value pairs and groups the values according to the key then output a set of intermediate grouped key-value pairs. The second function *Reduce* takes as input such intermediate grouped key-value pairs and aggregates the values according to each key.

Distributed computing in turn leads to high scalability, high efficiency, fault-tolerance, load balancing, and automatic parallelization. Distributed systems allow processing large amount of data on clusters of commodity hardware.

In 2005, Yahoo introduced Apache Hadoop as an open source MapReduce implementation [8]. The Hadoop Distributed File System (HDFS) is a disk-based file system that spans across multi nodes of a distributed system. HDFS automatically divides the files into blocks; each block has many replicas. Then, it and distributes these replicated blocks into all local disks nodes. After Hadoop becomes more popular and the most widely-used platform for distributed data processing some other open source software have been introduced to work on top of Hadoop as *Hadoop ecosystem*. The main components of Hadoop ecosystem are summarized as follows:

Apache ZooKeeper: ZooKeeper is software for providing group services , providing distributed synchronization services , maintains configuration , and naming registry for large distributed systems[9].

Apache Oozie: Oozie is a scalable, reliable system for scheduling workflow to manage Hadoop jobs. It is integrated with the other Hadoop ecosystem by supporting different kinds of jobs [10].

Apache Flume: Flume is a reliable distributed service for collecting and moving huge amounts of streaming log data from different sources to a central data store. Flume has a flexible and very simple architecture based on streaming data flows [11].

Apache Sqoop: Sqoop is software for transferring bulk of structured data between relational databases and Hadoop. It can be used for loading single table or executes SQL query. It can be used to put data from Hadoop into a relational database as export process. Sqoop has named after concatenating sql-hadoop [12].

Table 1. Summary of Hadoop ecosystem functionality

Hadoop Ecosystem	Main Function
HDFS	Object Storage (Unstructured data)
HBase	Table Storage (Structured data)
Cassandra	Table Storage (Structured data)
Hadoop Map Reduce (Yarn)	Distributed Data Processing
Hive	Query Processing
Pig	Data Analysis
Mahout	Machine Learning
Sqoop	Structured Data Connector (Export and Import Data)
Flume	Unstructured Data Transfer (ETL Tool)
Zoo Keeper	Coordination Services
Oozie	Work Flow Management and Scheduling

Apache Pig: Pig is software used to analyze big data sets. It consists of a high level language very similar to SQL to express data analysis programs. It allows Hadoop users to write complex MapReduce transformations using its Pig Latin language [13].

Apache HBase: HBase is a column oriented, distributed database designed to run on top of Hadoop Distributed File System. HBase is written in Java and modeled after Google's BigTable to provide BigTable-like capabilities for Hadoop. HBase is a NoSQL key-value data store that provides real-time read and write access to very large datasets [14].

Apache Cassandra: Cassandra is a distributed database management system designed to handle huge amounts of structured data through large commodity servers. It provides scalability and high availability with no single point of failure. Cassandra provides strong support for clusters extended over multiple datacentres [15].

Table 1 illustrates briefly the components of Hadoop Ecosystem.

In 2010, Apache Spark has been introduced at University of California, Berkeley by Matei Zaharia et al. [16]. Spark supports main-memory caching and possesses a loop-aware scheduler. Spark provides in-memory computing capabilities to deliver speed. Additionally, Spark as well as Hadoop implements the MapReduce paradigm and

is Java-based. These features enable users to deploy existing Hadoop application logic in Spark via many APIs for ease of development like Scala, Python, and Java to support a wide variety of applications. While MapReduce basically is designed to develop batch processing using disks as intermediate storage, Spark enables near-real time development by using memory for processing and sharing information efficiently [17].

Spark controls a set of libraries including MLlib, Spark SQL, GraphX, Spark Streaming, and Spark R. The basic data structure of Spark is Resilient Distributed Data sets (RDD) which is fault tolerant and works in parallel. Spark SQL is a Scala implementation of big data query processing system built on top of Spark. Spark SQL act as SQL Interface and programmatic interface to fill the gap between machine learning and analytics quires and considered as Spark structured data processing module which use a data frame [18, 19].

The key point of Spark SQL is data frames which is a distributed collection of rows with the same schema. Data frames are equivalent to table in traditional database systems. Data frames can be represented by using columnar data format which allow to access specific column without need to retrieve the whole data set. Spark SQL data frames is loaded into main memory instead of reading data from disk, so that the processing is more faster when compare with other big data processing systems, especially for iterative data processing. For all these reasons Spark SQL is considered as high-speed big data query processing system [19- 21].

3. Related work

During last decades, rough set theory is considered a very powerful data analysis technique and rough sets based methods have been successfully applied in knowledge discovery and data mining. Traditional rough sets based methods for knowledge discovery and acquisition has limitations to deal with the rapidly growing of data and the recently introduced MapReduce technique both of them pushed the researchers to pay great attention from researchers to integrate classical rough set theory with big data technologies and MapReduce technique to introduce new algorithms that overcome this rapidly growing of data. Zhang et al. has presented a parallel rough set based method for knowledge acquisition using MapReduce from big data. Their proposed model was developed on Hadoop platform which uses disk-based computation and its performance is limited compared with Spark in-memory computation platform [22, 23].

Guang et al. presented a new method for calculating reducts for condition attributes in decision table by using SQL [24]. Jemal et al. proposed integration between big data solutions and classical RDMS to achieve the benefits of each one and to introduce a new OLAP query process model integrated with MapReduce model [25]. Fernandez et al. provided a brief view on fuzzy systems for big data and the challenges and the opportunities facing this new framework. They introduced a fuzzy rule selection algorithm and emphasized the necessity of migrating programming towards Spark and Flink as newest frameworks of big data analysis [26-29]. In [30] Rochd and Hafidi presented a new approach for mining frequent itemsts in big data based on Hadoop by using N-List and they suggested HPrePostPlus Algorithm for mining frequent itemsts.

ROSETTA rough sets toolkit is considered as one of Benchmark data analysis tools which use rough set decision table classification and attributes reduction. However, it has a limitation when dealing with dataset size exceeds 30000 instances [31, 32]. Waikato Environment for Knowledge Analysis (WEKA) is also one of the most powerful tools for data analysis which has rough sets based decision table classifier but WEKA decision table classifier has some limitation with dealing with CSV files and big data [33]. Another powerful benchmark data analysis tool is KNIME Analytics Platform which has powerful interface dealing with most data sources especially CSV files. On the other hand, KNIME has a limitation with big datasets. This limitation can only be solved by increasing the CPU and RAM [34].

4. The proposed approach

In this paper we propose a novel approach for mining big data. The basic idea behind our proposed approach is based on the integrating fuzzy logic concepts, rough sets theory methodology of data mining, SQL, and big data technologies (Hadoop, Hive, Spark, and Spark-SQL) to generate consistent fuzzy decision rules from big data. Such an approach helps in extracting useful hidden knowledge in big data and supports the decision making process for analysing big data.

The proposed approach is achieved in three steps. The first step is to use fuzzy logic for fuzzifying numerical attributes in the given data sets to a set of granules, i.e. linguistic values. The

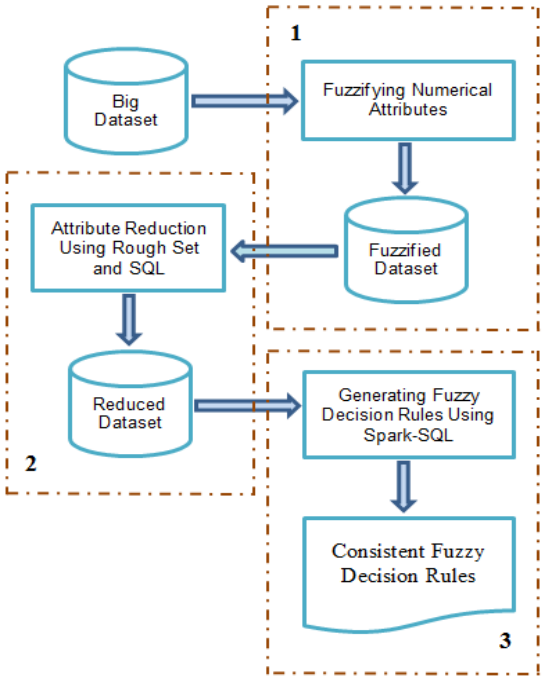

Figure.1 Structured diagram for our proposed approach

second step is the attribute reduction using rough set and SQL. The third step is the integration between rough sets data mining techniques with Spark-SQL. Fig. 1 shows the structured diagram for our proposed approach.

The above three step are explained in the following sub-sections.

4.1 Fuzzification of numerical attributes of information table

Fuzzification is the process of fuzzifying numerical values into linguistic terms, which is often used to reduce information overload in human decision making process. Using linguistic terms instead of numerical data helps in three ways; first linguistic value is more understandable to human, second to reduce the number of equivalence classes, third to minimize the number of extracted rules which helps decision maker to have a more clear view. It is worth to mention that fuzzification is a crucial step for our approach to enhance rule extraction process from the decision table.

4.2 Attribute reduction using rough set and SQL

In this section, we introduce two algorithms, inspired from [24], for rough set attributes reduction based on SQL. The first algorithm aims to eliminate fully functionally dependent attributes as follows:

Algorithm 1: Eliminating Attributes with full functional dependency

Input : S = (U, A) is an information system

Output : reduced condition attributes

Method :

1- Initial condition attributes A
2- For i =1 to | condition attributes |
3- Calculate Card (a_i)
4- For j = 1 to | condition attributes |
5- If a_i <> a_j
6- Calculate Card($\Pi (a_i + a_j)$
7- If Card (a_i) = Card($\Pi (a_i + a_j)$)
8- and Card (a_i) > 3 then
9- A = A – { a_j }
10- End If
11- End If
12- Next j
13- Next i

The SQL Implementation of Card (a_i) :

```
SELECT    COUNT( * )
FROM      ( SELECT    ai
            FROM      T
            GROUP BY  ai )
```

The SQL Implementation of Card (a_i , a_j) :

```
SELECT    COUNT( * )
FROM      ( SELECT    ai , aj
            FROM      T
            GROUP BY ai , aj )
```

The second algorithm aims to find out all reducts as follows:

Algorithm 2: Finding Reducts

Input : S = (U, A) is an information system

Output : a set of all attribute reductions REDU

Method :

1- REDU Set = ∅
2- Run Algorithm 1 to eliminate Attributes with complete functional dependency
3- Calculate of the power set of reduced condition attributes = {P1, P2,…..}

4- Calculate cardinality of the attribute set = Card (A)
5- For i = 1 to | Power Set |
6- calculate cardinality of P_i = Card (P_i)
7- If Card (P_i) = Card (A) then
8- REDU Set = REDU Set ∪ { P_i }
9- Power Set = Power Set – { P_i}
10- For j = i+1 to | Power Set |
11- If $P_i \subseteq P_j$ then
12- Power Set = Power Set – {Pj}
13- End if
14- Next j
15- End if
16- Next i

The SQL Implementation of Card (A) :

```
SELECT  DISTINCT COUNT(*)
FROM    (SELECT    A
         FROM      T
         GROUP BY  A  )
```

The SQL Implementation of Card (ai , aj) :

```
SELECT  DISTINCT COUNT(*)
FROM    (SELECT     Pi
         FROM       T
         GROUP BY   Pi )
```

4.3 Extracting fuzzy classification rules

After fuzzifying original decision table and determining the reducts attributes, we can get fuzzy decisions rules which help decision maker to take the proper decision. We proposed a novel algorithm for extracting consistent fuzzy rules from rough set information table using SPARK-SQL and SCALA statements as shown in algorithm 3.

Algorithm 3: Generating consistent Fuzzy Decision Rules

Input : fuzzy decision table; REDU , MinSupport
Output : a set of all consistent fuzzy decisions rules;
Method :

1- For i = 1 to | REDU |
2- MinSupport = Min No. of rule occurrence
3- Create TmpReductTbl
4- Create TmpReductPlusDecisionTbl
5- Rules = Ø
6- For j = 1 to | TmpReductTbl |
7- If Card (C) = Card (C+D)
8- If Support >= MinSupport
9- Rules= Rules+EquivClassPlusD
10- End if
11- End if
12- Next j
13- Next i

We have Implemented Algorithm 3 using Spark-SQL and SCALA statements for extracting fuzzy rules through three main steps as follows:

1- Create temp table for equivalence classes for current reduct

```
val  MyRdd = sc.textFile("/MyPath/MyDataSet.csv")
val MyDataSetRdd = MyRdd.map { line => val cols = line.split(";") ( cols(0) , cols(1) , cols(2) , cols(n) ) }
val MyDataSetDF  = MyDataSetRdd.toDF( "Col1Name"     , " Col2Name " , " Col3Name " , " ColnName "  )
MyDataSetDF.registerTempTable("TempReductTable")
val TempReductTable = sql(" SELECT      CurrentReductAttributes ,  COUNT (*) As Support
FROM  TmpReductTbl
GROUP BY CurrentReductAttributes ")
TempReductTable.write.mode("overwrite").saveAsTable("TmpReductTbl")
```

2- Create temp table for equivalence classes for current reduct plus decision attribute

```
MyDataSetDF.registerTempTable("TmpReductPlusDecisionTbl ")
val  TmpReductPlusDecisionTbl =
sql("SELECT CurrentReductAttributes , D ,  COUNT (*) As Support
FROM TmpReductPlusDecisionTbl
GROUP BY CurrentReductAttributes,D")
```

```
TmpReductPlusDecisionTbl.write.mode("overwrite").saveAsTable("TmpReductPlusDecisionTbl")
```

3- Get consistent decision rule for each equivalence class with given minimum support

```
val MyDecisionRules = sql(" SELECT concat(' If X1 = ' , A.X1 ) , concat(' And X2= ' , A.X2 ) , concat(' And Xn= ' , A.Xn ) ,  concat(' Then D = ' , A.D ),  A.Support
FROM TmpReductPlusDecisionTbl A , TmpReductTbl B
WHERE A.X1 = B. X1
AND    A.X2 = B.X2
AND   … A.Xn = B.Xn
AND   A.Support = B.Support
AND   A.Support >= @@MinSupport
ORDER BY Support Desc")
MyRules.rdd.coalesce(1,true).saveAsTextFile(" MyPath/MyRules")
```

The previous SQL Condition (`A.Support = B.Support`) guarantees the consistency of the extracted rules. In other words, the extracted classification rules have classification accuracy 100%, So that we don't need to construct confusion matrix to calculate the classification accuracy.

5. Evaluation

To evaluate our proposed approach, we used some benchmark datasets (UCI Datasets) downloaded from the UC Irvine Machine Learning Repository [35]. All experiments were conducted using the benchmark applications ROSETTA rough sets toolkit and WEKA decision table classifier through KNIME Analytics Platform.

Our proposed approach is implemented on top of SPARK platform using Virtual Machine with operating system Ubuntu 16.04, CPU Intel i7 2 Cores, 4.3 GB RAM and Big Data Platform is Hadoop 2.7 – Spark 2.0 – Hive 2.0 – Scala 2.10.4 [8, 36-38] . We evaluate our work with Weka Decision Table Classifier through KNIME platform as it is more flexible when dealing with CSV files, using Physical Machine with Windows 7 CPU Intel corei7 2.67 GHz 4-cores 64-bit with 8 GB RAM.

Table 2 shows the comparisons between the proposed approach for attributes reduction, ROSETTA's genetic algorithm and WEKA 3.7 decision table classifier

Table 2. UCI datasets and number of extracted reducts

Dataset	Features	Instances	No. of Reducts		
			Rosetta	Weka	Proposed Approach
Zoo	16	101	7	1	5
Yeast	8	1484	1	1	1
Abalone	8	4177	16	1	15
Car	6	1728	1	1	1
Glass	9	214	12	1	11
Adult	14	32561	1	1	1
Breast Cancer	9	286	1	1	1
Iris	4	150	1	1	1
Nursery	8	12960	1	1	1
Balloons	4	20	1	1	1

It should be noted that instances mentioned in Table 2 are pre-processed to remove redundancies. So, for example, the adult data set is reduced to be 14072 instances. Also, we noticed that WEKA decision table classifier generates only one reduct set while our proposed approach and ROSETTA's genetic algorithm generate all possible reduct set which give more variety of extracted rules.

Table 3 illustrates the comparison between the proposed approach for rules extraction, ROSETTA's genetic algorithm and WEKA decision table classifier.

Table 3. UCI datasets and number of extracted rules

Dataset	No. of Rules		
	Rosetta	Weka	Proposed Approach
Zoo	413	14	295
Yeast	1453	109	1453
Abalone	66827	122	62655
Car	1728	432	1728
Glass	2556	37	2343
Adult	12999	427	11926
Breast Cancer	272	33	266
Iris	147	3	147
Nursery	12960	810	12960
Balloons	16	4	16

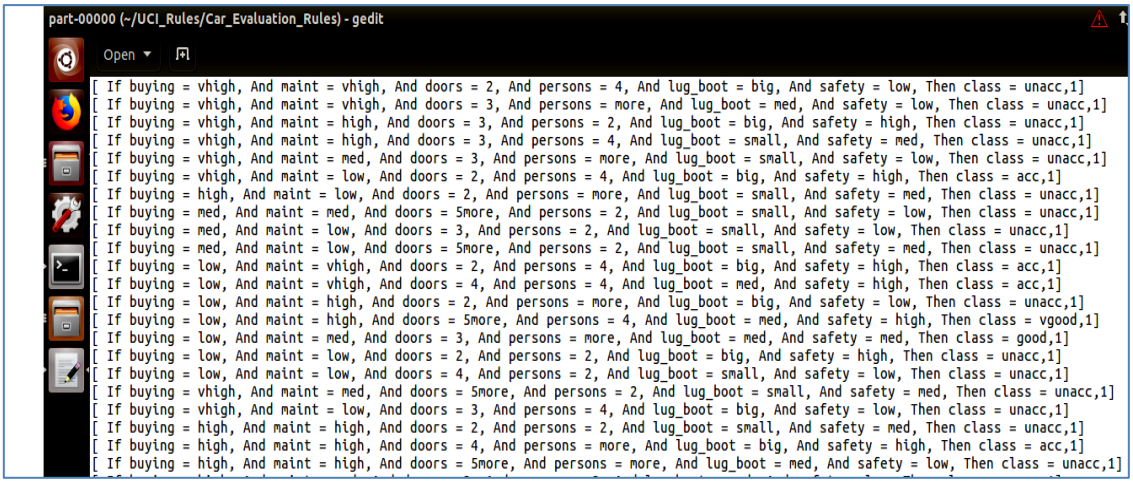

Figure.2 Screenshot of sample of extracted rules for car dataset

Fig. 2 shows a screenshot of a sample of the extracted decision rules of Car dataset using our proposed approach.

Table 4 illustrates the comparison of the classification accuracy between the proposed approach, ROSETTA's genetic algorithm and WEKA decision table classifier. It is noted that, our proposed approach as well as ROSETTA share the high classification accuracy for the extracted rules.

In order to evaluate our proposed approach with big datasets, we used oversampled real datasets for Egyptian Investment Companies, which describe the companies' profile. Each instance represents one company data such as: amount of shares, types of shares (Egyptian shares - Foreign shares), location (governorate - canton), activities category (sector - subsector), legal form, working status, expansion status. Such a huge dataset is manipulated in two different groups of datasets as follows:

- First datasets group contain six attributes divided as five conditional attributes namely, *Total_Shares* (numeric), *Sector* (nominal), *Capital_Flow*(nominal), *Legal_Form*(nominal), *Governorate* (nominal), and one decision attribute namely, *Project_Status* (nominal). The reduct contains all condition attributes. We divide this group into three granules: Crisp, fuzzified with three linguistic values (*Small, Medium, Big*), and fuzzified with five linguistic values (*Tiny, Small, Medium, Big, Huge*). We used fuzzy triangular membership function during fuzzification process as one of the most commonly used fuzzy membership functions. Table 5 illustrates the properties of the first datasets group with its twelve oversampled versions.
- Second datasets group contain thirteen Attributes divided as twelve conditional attributes namely, *Egy_Shares* (numeric), *Forign_Shares* (numeric), *Total_Shares* (numeric), *Expansions* (numeric), *Project_Status* (nominal), *Sector* (nominal), *Sub_Sector* (nominal), *Capital_Flow* (nominal), *Legal_Form* (nominal), *Operatoinal_Satus* (nominal), *Governorate* (nominal), *Canton* (nominal), and one decision attribute namely, *Expansion_Status* (nominal). The reduct contains only ten conditional attributes with elimination of the two attributes *Sector* and *Canton*. We divide this group also into three granules: Crisp, fuzzified with three linguistic values (*Small, Medium, Big*), and fuzzified with five linguistic values (*Tiny, Small, Medium, Big, Huge*). We used also fuzzy triangular membership function during Fuzzification process. Table 6 illustrates the properties of the second datasets group with its twelve oversampled versions.

Table 4. UCI datasets and accuracy of extracted rules

Dataset	Rosetta (RHS Accuracy)	Weka (Merit of best subset)	Proposed Approach (RHS Accuracy)
Zoo	1	0.27	1
Yeast	1	58.29	1
Abalone	1	2.37	1
Car	1	95.25	1
Glass	1	1.17	1
Adult	1	82.63	1
Breast Cancer	1	79.72	1
Iris	1	96	1
Nursery	1	94.82	1
Balloons	1	100	1

Table 5. The first group datasets (6 Attributes)

NO.	Dataset	No. of Records	Size (Mbytes)
1	Crisp_1M_6C	1,000,000	73
	Fuzzy_1M_3V_6C	1,000,000	75
	Fuzzy_1M_5V_6C	1,000,000	74
2	Crisp_2M_6C	2,000,000	147
	Fuzzy_2M_3V_6C	2,000,000	150
	Fuzzy_2M_5V_6C	2,000,000	149
3	Crisp_3M_6C	3,000,000	221
	Fuzzy_3M_3V_6C	3,000,000	226
	Fuzzy_3M_5V_6C	3,000,000	225
4	Crisp_4M_6C	4,000,000	294
	Fuzzy_4M_3V_6C	4,000,000	301
	Fuzzy_4M_5V_6C	4,000,000	299
5	Crisp_5M_6C	5,000,000	368
	Fuzzy_5M_3V_6C	5,000,000	377
	Fuzzy_5M_5V_6C	5,000,000	373
6	Crisp_10M_6C	10,000,000	752
	Fuzzy_10M_3V_6C	10,000,000	770
	Fuzzy_10M_5V_6C	10,000,000	763
7	Crisp_15M_6C	15,000,000	1,120
	Fuzzy_15M_3V_6C	15,000,000	1,145
	Fuzzy_15M_5V_6C	15,000,000	1,134
8	Crisp_20M_6C	20,000,000	1,487
	Fuzzy_20M_3V_6C	20,000,000	1,522
	Fuzzy_20M_5V_6C	20,000,000	1,507
9	*Crisp_30M_6C	30,000,000	2,240
	*Fuzzy_30M_3V_6C	30,000,000	2,292
	*Fuzzy_30M_5V_6C	30,000,000	2,269
10	*Crisp_40M_6C	40,000,000	2,986
	*Fuzzy_40M_3V_6C	40,000,000	3,055
	*Fuzzy_40M_5V_6C	40,000,000	3,025
11	*Crisp_50M_6C	50,000,000	3,727
	*Fuzzy_50M_3V_6C	50,000,000	3,814
	*Fuzzy_50M_5V_6C	50,000,000	3,776
12	*Crisp_60M_6C	60,000,000	4,480
	*Fuzzy_60M_3V_6C	60,000,000	4,584
	*Fuzzy_60M_5V_6C	60,000,000	4,539

* KNIME (WEKA decision table classifier) conducted on 32GB RAM- CPU 6 Cores

Table 6. The second group datasets (13 Attributes)

NO.	Dataset	No. of Records	Size (Mbytes)
1	Crisp_1M_13C	1,000,000	142
	Fuzzy_1M_3V_13C	1,000,000	153
	Fuzzy_1M_5V_13C	1,000,000	150
2	Crisp_2M_13C	2,000,000	284
	Fuzzy_2M_3V_13C	2,000,000	306
	Fuzzy_2M_5V_13C	2,000,000	299
3	Crisp_3M_13C	3,000,000	425
	Fuzzy_3M_3V_13C	3,000,000	459
	Fuzzy_3M_5V_13C	3,000,000	448
4	Crisp_4M_13C	4,000,000	566
	Fuzzy_4M_3V_13C	4,000,000	610
	Fuzzy_4M_5V_13C	4,000,000	597
5	Crisp_5M_13C	5,000,000	631
	Fuzzy_5M_3V_13C	5,000,000	763
	Fuzzy_5M_5V_13C	5,000,000	747
6	*Crisp_10M_13C	10,000,000	1,275
	*Fuzzy_10M_3V_13C	10,000,000	1,539
	*Fuzzy_10M_5V_13C	10,000,000	1,505
7	*Crisp_15M_13C	15,000,000	1,951
	*Fuzzy_15M_3V_13C	15,000,000	2,344
	*Fuzzy_15M_5V_13C	15,000,000	2,293
8	*Crisp_20M_13C	20,000,000	2,582
	*Fuzzy_20M_3V_13C	20,000,000	3,107
	*Fuzzy_20M_5V_13C	20,000,000	3,039
9	*Crisp_30M_13C	30,000,000	3,857
	*Fuzzy_30M_3V_13C	30,000,000	4,646
	*Fuzzy_30M_5V_13C	30,000,000	4,544
10	*Crisp_40M_13C	40,000,000	5,757
	*Fuzzy_40M_3V_13C	40,000,000	6,190
	*Fuzzy_40M_5V_13C	40,000,000	6,054
11	**Crisp_50M_13C	50,000,000	6,440
	**Fuzzy_50M_3V_13C	50,000,000	7,755
	**Fuzzy_50M_5V_13C	50,000,000	7,585
12	**Crisp_60M_13C	60,000,000	8,643
	**Fuzzy_60M_3V_13C	60,000,000	9,294
	**Fuzzy_60M_5V_13C	60,000,000	9,090

* KNIME (WEKA decision table classifier) conducted on 32GB RAM- CPU 6 Cores

** KNIME (WEKA decision table classifier) failed on 32GB RAM- CPU 6 Cores

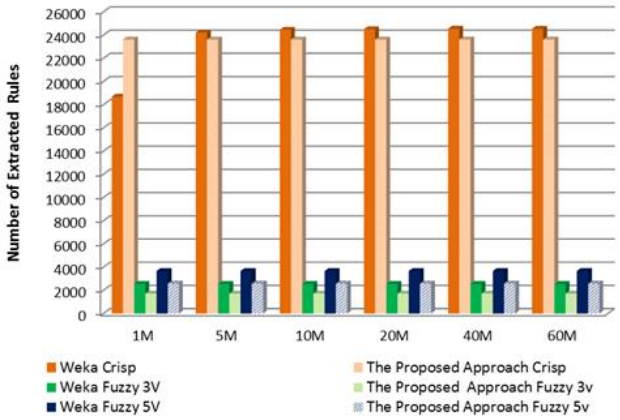

Figure.3 No. of extracted rules for six attributes datasets

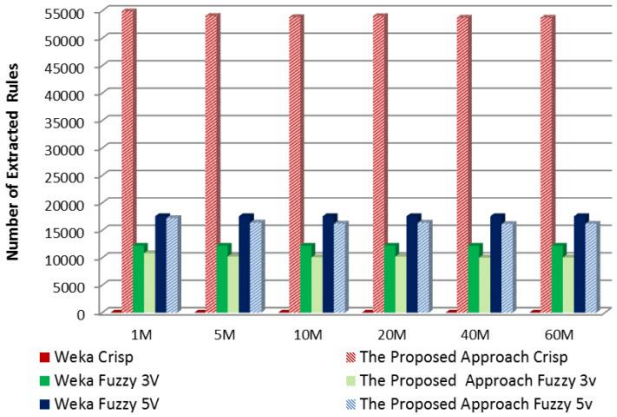

Figure.4 No. of extracted rules for thirteen attributes datasets

We have evaluated our proposed approach for these big datasets with only WEKA decision table classifier through KNIME platform because ROSETTA is limited to 30000 instances only. During the reduction process for all such datasets both of our proposed approach and WEKA decision table classifier generate only one reduct set.

In the first group datasets (six attributes), our proposed approach generates less number of consistent decision rules compared with WEKA decision table classifier rules as shown in Fig. 3 . In addition, Fig. 3 shows clearly that the numbers of extracted rules in both fuzzified datasets are very small compared with crisp datasets. Moreover, the number of extracted rules of datasets fuzzified with three linguistic values is less than number of rules extracted from datasets fuzzified with five linguistic values

For the second group datasets (thirteen attributes), Fig. 4 shows clearly the big gap between the number of extracted rules for crisp datasets and both fuzzified datasets. In addition, the proposed approach generates less number of consistent decision rules in fuzzified datasets compared with WEKA decision table classifier rules. On the other hand, in crisp datasets WEKA generates only two rules which mean our proposed approach can handle both crisp and fuzzified datasets for these big datasets with large number of features.

Thus, the experimental results show that the proposed approach can improve the classification accuracy for the extracted classification rules compared with WEKA decision table classifier in all datasets groups because we exclude inconsistent rules and generate only the consistent rules as we have mentioned in section 4.3. The classification accuracy is 100%.

The experimental analysis for the performance of our proposed approach showed that the proposed approach can handle very big datasets easier than WEKA decision table classifier regarding to execution cost and computing cost. The execution time is reduced with 90-95% in addition the proposed approach reduces the needed computing resources. This improvement of performance is mainly due to the fact that the proposed approach uses Spark in-memory platform plus using SQL statements for generating consistent classification rules

It should be mentioned that, Table 5 illustrates that KNIME Platform using WEKA decision table classifier failed to execute more than twenty million records on the first test environment with CPU Intel corei7 2.67 GHz 4-cores 64-bit with 8 GB RAM, then we conducted the rest of experiments on the second test environment CPU Intel Xeon E5-264 2.40 GHz 6-cores 64-bit, with 32 GB RAM, see the note below Table 5. In addition, Table 6 shows that KNIME also failed with datasets size fifty and sixty million and required more hardware resources than 32GB RAM. Regardless of such exceptional cases, all our experiments conducted successfully on Ubuntu 16.04, CPU Intel i7 2 Cores, with 4.3 GB RAM.

For a performance comparisons based on the execution time costs, Figs. 5 to 8 illustrate the big gap between proposed approach and WEKA decision table classifier.

Fig. 5 shows the comparison of execution time between our proposed approach and WEKA decision table classifier for first group datasets fuzzified with three linguistic values. It is noted that for small and medium datasets size (one million up to five million records) the execution cost gap is not that big, but in biggest datasets size (ten million up to sixty million records) the execution time is greatly reduced with 90% as shown Fig. 5.

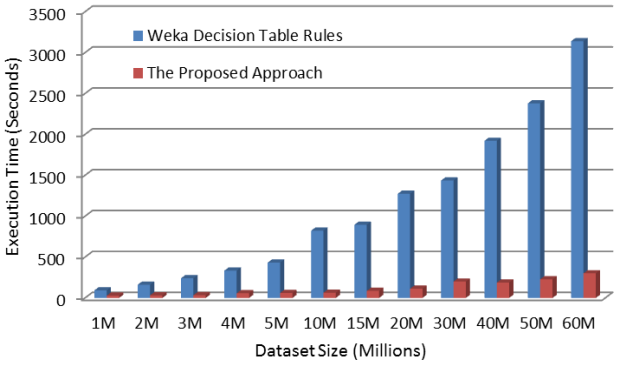

Figure.5 Execution time for six attributes datasets fuzzified with three linguistic values

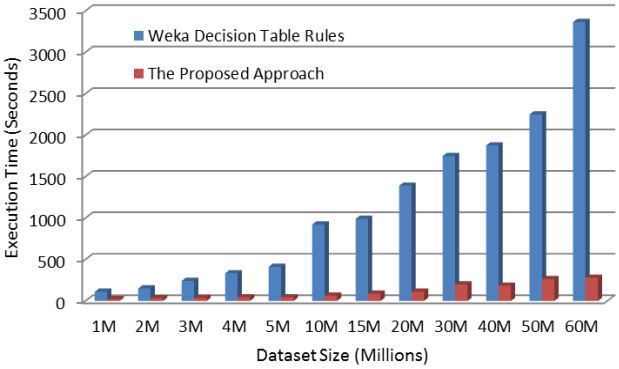

Figure.6 Execution time for six attributes datasets fuzzified with five linguistic values

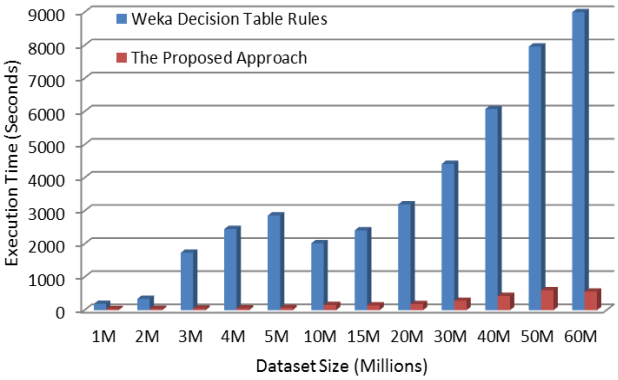

Figure.7 Execution time for thirteen attributes datasets fuzzified with three linguistic values

Fig. 6 shows the comparison of execution time for first group datasets fuzzified with five linguistic values. We can also see clearly the difference of execution cost between the proposed approach and WEKA decision table classifier especially in very big datasets. The cost of execution time is saved up to 90%.

Figs. 5 and 6 clearly illustrate the big gap in execution time between our proposed approach and WEKA decision table classifier through KNIME Platform. In addition, WEKA decision table classifier through KNIME Platform needed more

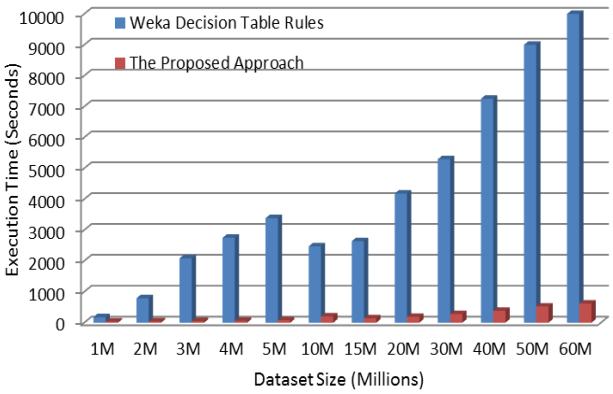

Figure.8 Execution time for thirteen attributes datasets fuzzified with five linguistic values

hardware resources for some datasets , see the note below Table 5.

Figs. 7 and 8 show the comparison of execution time between our proposed approach and WEKA decision table classifier for thirteen attributes datasets fuzzified with three linguistic values and five linguistic values respectively. It is clearly observed from Figs. 7 and 8 that our proposed approach has a much better performance than WEKA. In addition, WEKA decision table classifier through KNIME Platform failed to run and needed more hardware resources for fifty and sixty million records data sets size as given in the note below Table 6. We can also clearly see that the cost of execution time is saved up to 95% for the second group of datasets with larger number of features. This ensures that the proposed approach performance is quite suitable for dealing big datasets with larger numbers of features.

Figs. 9 and 10 illustrate sample of extracted fuzzy rules for both datasets groups. It is noted that these rules are more readable and more human understandable compared with Weka extracted rules given in Fig. 11. On the other hand, Fig. 11 shows screenshot of KNIME Analytics Platform using Weka Decision table classifier 3.7 and sample of its extracted rules for six attributes dataset with one million instances.

6. Conclusion

This paper presents a novel approach for mining fuzzy classification rules from big data sets to enhance knowledge extraction decision making process. The proposed approach, integrates fuzzy sets concepts, especially fuzzy linguistic values, and rough sets data analysis concepts, especially decision table, equivalence classes, and reducts with new technologies of big data especially Spark,

Figure.9 Screenshot of sample of our approach extracted rules for six attributes datasets

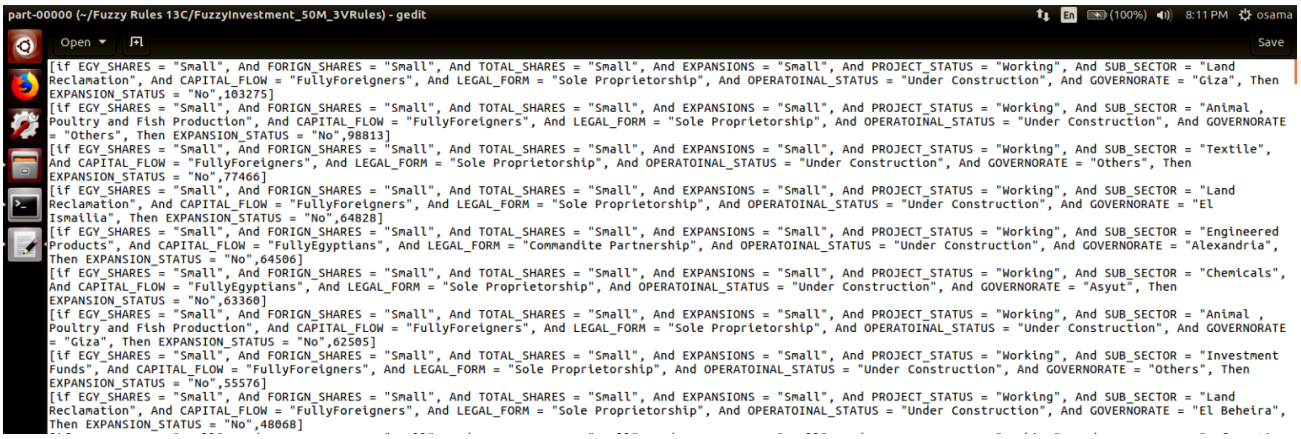

Figure.10 Screenshot of sample of our approach extracted rules for thirteen attributes dataset

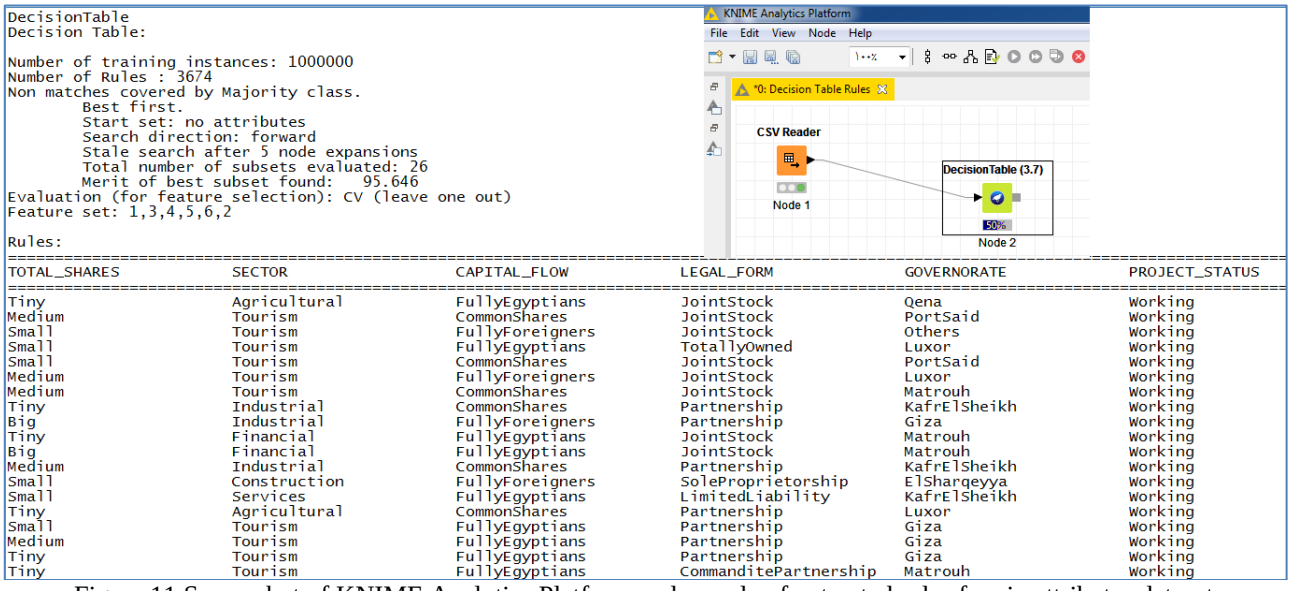

Figure.11 Screenshot of KNIME Analytics Platform and sample of extracted rules for six attributes dataset

Spark-SQL, and Hive. The proposed approach takes advantages of both the high performance of Spark in data analysis and granular computing.

The proposed approach is tested against benchmark data analysis tools ROSSETA Rough Set Toolkit for Analysis of Data and WEKA decision table classifier through KNIME Analytics Platform. Several experiments have been carried out with two categories of datasets, first some benchmark UCI datasets, and the second is oversampled real data for Egyptian Investment

Companies profile divided to two groups based on the number of conditional attributes.

The results showed that the proposed approach works extremely faster with big datasets and outperforms the existing data analysis tools. According to the experimental results the execution time of big datasets enhanced by 90% for first group of big datasets and 95% for the second group of big datasets. The classification accuracy for our proposed approach is 100% which generate more efficient, more accurate, and more human understandable fuzzy classification rules with less cost of execution time.

References

[1] Koliopoulos, P. Yiapanis, F. Tekiner, G. Nenadic, and J. Keane, "A Parallel Distributed Weka Framework for Big Data Mining using Spark", In: *Proc. of IEEE International Congress on Big Data*, pp. 9–16, 2015.

[2] J. Yao, A. Vasilakos, and W. Pedrycz, "Granular computing: perspectives and challenges", *IEEE Transactions on Cybernetics*, Vol. 43, No. 6, pp. 1977–1989, 2013.

[3] Y. Yao, "Granular computing for data mining", In: *Proc. of the SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, pp. 1–12, 2006.

[4] L. Zadeh, "Towards a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic", *Fuzzy Sets and Systems*, Vol. 90, pp.111-127, 1997.

[5] T. Lin, "From rough sets and neighborhood systems to information granulation and computing in words", In: *Proc. of European Congress on Intelligent Techniques and Soft Computing*, pp. 1602-1607, 1997.

[6] Z. Pawlak, "Rough sets", *International Journal of Computer and Information Sciences*, Vol. 11, No. 5, pp. 341–356, 1982.

[7] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters", In: *Proc. of the 6th conference on Symposium on Operating Systems Design & Implementation*, pp. 137-150, 2004.

[8] Apache Hadoop: https://hadoop.apache.org/

[9] Zoo Keeper: http://zookeeper.apache.org/

[10] Apache Oozie: https://oozie.apache.org/

[11] Apache Flume: https://flume.apache.org/

[12] Apache Sqoop: http://sqoop.apache.org/

[13] Apache PIG: http://pig.apache.org/

[14] Apache HBase: https://hbase.apache.org/

[15] Cassandra: http://cassandra.apache.org/

[16] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets", In: *Proc. of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2010.

[17] N. Katsipoulakis, Y. Tian, B. Reinwald, and H. Pirahesh, "A Generic Solution to Integrate SQL and Analytics for Big Data", In*: Proc. of the 18th International Conference on Extending Database Technology*, pp. 671–676, 2015.

[18] W. Alkowaileet, S. Alsubaiee, M. Carey, T. Westmann, and Y. Bu, "Large-scale complex analytics on semi-structured datasets using asterixDB and spark", In*: Proc. of the VLDB Endowment*, Vol. 9, No.13, pp.1585-1588, 2016

[19] K. Park and L. Peng, "A Design of High-speed Big Data Query Processing System for Social Data Analysis: Using Spark SQL", *International Journal of Applied Engineering Research*, Vol. 11, No. 14, pp. 8221-8225, 2016.

[20] K. Naacke, O. Curé, and B. Amann, "SPARQL Query Processing with Apache Spark", *arXiv:1604.08903 [cs.DB]*, pp. 10–23, 2016 .

[21] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, and M. Zaharia, "Spark sql: Relational data processing in spark", In: *Proc. of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1383-1394, 2015.

[22] J. Zhang, T. Li, and Y. Pan, "Parallel rough set based knowledge acquisition using MapReduce from big data", In: *Proc. of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pp. 20–27, 2012.

[23] J. Zhang, J. Wong, T. Li, and Y. Pan, "A comparison of parallel large-scale knowledge acquisition using rough set theory on different MapReduce runtime systems", *Elsevier International Journal of Approximate Reasoning*, Vol. 55 , No. 3 , pp. 896–907, 2014.

[24] J. Guang, C. Fu-yuan, Z. Yi-chi, and G. Jia-wei, "Method of Attribute Reduction of Rough Set Based on SQL", *Journal of Computer Engineering*, Vol. 34, No.11, pp. 67-71, 2008.

[25] D. Jemal, R. Faiz, A. Boukorca, and L. Bellatreche, "MapReduce-DBMS: An Integration Model for Big Data Management and Optimization", *Database and Expert*

Systems Applications. Globe 2015, DEXA 2015, Lecture Notes in Computer Science, Vol. 9262, pp. 430-439, 2015.
[26] A. Fernandez, C. Carmona, M. del Jesus, and F. Herrera, "A view on fuzzy systems for big data: progress and opportunities", *International Journal of Computational Intelligence Systems*, Vol. 9, No. 1, pp. 69–80, 2016.
[27] Fernandez, E. Almansa, and F. Herrera, "Chi-Spark-RS: An spark-built evolutionary fuzzy rule selection algorithm in imbalanced classification for big data problems", In: *Proc. of 2017 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE 2017*, pp. 1-6, 2017.
[28] Fernandez, S. Rio, A. Bawakid, and F. Herrera, "Fuzzy rule based classification systems for big data with MapReduce: Granularity analysis", *Advances in Data Analysis and Classification*, Vol.11 No.4 , pp. 711-730 , 2017.
[29] Fernandez, A. Altalhi, S. Alshomrani, and F. Herrera, "Why linguistic fuzzy rule based classification systems perform well in big data applications?", *International Journal of Computational Intelligence Systems*, Vol. 10, pp. 1211–1225, 2017.
[30] Y. Rochd and I. Hafidi, "Performance Improvement of PrePost Algorithm Based on Hadoop for Big Data", *International Journal of Intelligent Engineering and Systems*, Vol.11, No.5, pp. 226-235, 2018
[31] ROSETTA : http://bioinf.icm.uu.se/rosetta/
[32] A. Øhrn and J. Komorowski, "ROSETTA: A rough set toolkit for analysis of data", In: *Proc. of the Third International Joint Conference on Information Sciences*, Vol. 3, pp. 403–407, 1997.
[33] Weka : https://www.cs.waikato.ac.nz/ml/weka/
[34] KNIME : https://www.knime.com
[35] UC Irvine Machine Learning Repository http://archive.ics.uci.edu/ml/index.php
[36] Apache Spark : https://spark.apache.org/
[37] Scala : https://www.scala-lang.org/download/
[38] Apache Hive : http://hive.apache.org