PARALLEL ASSOCIATION RULE MINING ON SEMANTIC AND BIG IoT DATA

AMAL ALSAEH

JULY 2018

PARALLEL ASSOCIATION RULE MINING ON SEMANTIC AND BIG IoT DATA

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF

ÇANKAYA UNIVERSITY

BY

AMAL ALSAEH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER ENGINEERING DEPARTMENT

July 2018

Title of the Thesis: **Association Rule Mining on Semantic and Big IoT Data.**

Submitted by: **Amal ALsaeh.**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

[illegible]

Prof. Dr. Can ÇOĞUN [illegible]
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Erdoğan DOĞDU
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Erdoğan DOĞDU
Supervisor

Examination Date 5/7/2018

Examining Committee Members

Assoc. Prof. Dr. Ahmet Murat Özbayoğlu (TOBB University) [illegible]

Prof. Dr. Erdoğan DOĞDU (Çankaya University) [illegible]

Assoc.Prof. Dr. Reza ZARE (Çankaya University) R. Hassanpour

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name	: Amal ALSAEH
Signature	: [illegible]
Date	: 2/8/2018

ABSTRACT

PARALLEL ASSOCIATION RULE MINING ON SEMANTIC AND BIG IOT DATA

AMAL ALSAEH

M.Sc., Department of Computer Engineering

Supervisor: Prof. Dr. Erdoğan DOĞDU

July 2018

Association Rule Mining (ARM) is an important machine learning technique because it can find associations or relationships between data items in large datasets. Different association rule algorithms have been studied by many researchers in traditional transactional data sets in which, all items have a unique relationship such as, 'buy'. In recent years, researchers have shown an increased interest in extracting association rules from semantic graphs (such as RDF datasets) instead of traditional data. On the other hand, in the field of the Internet of Things (IoT) and in many different domains, semantic data is daily increasing in large volumes. Therefore, we need scalable solutions to utilize IoT data for intelligent solutions. In this thesis, we studied the utilization of parallelized FP-growth algorithm on several different sematic IoT datasets from different domains, such as weather, traffic, and medicine. The results show that the scalable execution of semantic ARM algorithms produces the semantic association rules much faster.

Keywords: Association rule mining, Scalable association rule mining, Semantic data, RDF, IoT, Machine learning, MapReduce.

ÖZ

SEMANTIK VE BÜYÜK IOT VERISI ÜZERINDE PARALEL BIRLIKTELIK KURALI MADENCILIĞI

AMAL ALSAEH

Yüksek Lisans, bilgisayar mühendisliği Anabilim Dalý

Tez Yöneticisi: Prof. Dr. Erdoğan DOĞDU

Temmuz 2018

Birliktelik Kuralı Madenciliği (ARM) önemli bir makine öğrenme tekniğidir çünkü büyük veri kümelerindeki veri ögeleri arasında ilişkileri bulabilir. Farklı birliktelik kuralı algoritmaları, tüm ögelerin "satın alma" gibi benzersiz bir ilişkiye sahip olduğu geleneksel işlem veri kümelerinde birçok araştırmacı tarafından incelenmiştir. Son yıllarda araştırmacılar, geleneksel veriler yerine, anlamsal çizgelerden (RDF veri kümeleri gibi) birliktelik kurallarının çıkarılmasına artan bir ilgi göstermektedirler. Öte yandan, Nesnelerin İnterneti (IoT) alanında ve birçok farklı alanda, semantik veriler günlük olarak büyük miktarlarda artmaktadır. Bu nedenle, akıllı çözümler için IoT verilerini kullanmak üzere ölçeklenebilir çözümlere ihtiyacımız var. Bu tez çalışmasında, hava durumu, trafik ve tıp gibi farklı alanlardan birçok farklı sematik IoT veri kümesinde paralelleştirilmiş FP-büyüme algoritmasının kullanımını inceledik. Sonuçlar, semantik ARM algoritmalarının ölçeklenebilir uygulamasının semantik birliktelik kurallarını çok daha hızlı bulduğunu göstermektedir.

Anahtar kelimeler: Birliktelik kuralı madenciliği, ölçeklenebilir birliktelik kuralı madenciliği, semantik veri, RDF, IoT, makina öğrenmesi, MapReduce.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Prof. Dr. **Erdogan DOGDU** for his supervision, special guidance, suggestions, and encouragement through the development of this thesis.

Thanks to my parents, for encouraging and praying to me.

Thanks to my children, **Rayan**, **Alsaeh**, **Mohammad** and **Mais**, for sharing the difficulties with me.

Finally, I would like to thank my wonderful husband, **Ahmed Alsaeh** who was the best companion in Turkey for his patience, understanding, and moral support. Words cannot express my gratitude.

LIST OF ABBREVIATIONS

ARM	Association Rule Mining
Bnode	Blank Node
DECLAT	Diffset Equivalence Class Transformation
ECLAT	Equivalence Class Transformation
IoT	Internet of Things
LSM	Linked Stream Middleware
PFP	Parallel FP-growth
RDD	Resilient Distributed Datasets
RDF	Resource Description Framework
SAG	Semantic Association Generator
SANSA	Semantic Analytics Stack
SPARQL	Semantic Protocol And RDF Query Language
SQL	Structured Query Language
SWApriori	Semantic Web Apriori
SWARM	Semantic Web Association Rule Mining
TURTLE	Terse RDF Triple Language
UDF	User Defined Function
URI	Uniform Resource Identifier

TABLE OF CONTETNS

STATEMENT OF NON-PLAGIARISM PAGE iii

ABSTRACT iv

ÖZ v

ACKNOWLEDGEMENTS vi

LIST OF ABBREVIATIONS vii

TABLE OF CONTETNS viii

LIST OF TABLES xiii

LIST OF FIGURES xv

LIST OF CODES xvii

CHAPTER 1 1

1 Introduction 1

1.1 Overview 1

1.2 Internet of Things and Big Data 2

1.2.1 The Semantic Web 3

1.2.2 MapReduce 4

1.2.3 Apache Spark 5

1.2.4 Association Rule Mining (ARM) on IoT data 7

1.3 Objective 8

1.4 Organization of the Thesis 9

CHAPTER 2 10

2 Association Rule Mining 10

2.1 Definition........10
2.2 Apriori algorithm........11
2.2.1 Apriori algorithm example........12
2.2.2 Limitations of Apriori algorithm........15
2.2.3 Improving the efficiency of Apriori algorithm........15
2.3 FP-growth algorithm........16
2.3.1 Advantages of FP-growth algorithm........19
2.3.2 Disadvantages of FP-growth algorithm........19
2.4 Mining frequent patterns by using vertical data format........20
2.4.1 ECLAT algorithm........20
2.4.2 DECLAT algorithm (Mohammed J Zaki & Gouda, 2003)........22
2.4.3 Comparison between ECLAT and DECLAT algorithms:........22
2.5 CHAPTER SUMMARY........22
CHAPTER 3........24
3 Semantic Association Rule Mining........24
3.1 SWARM algorithm........24
3.2 SAG algorithm........26
3.3 SWApriori........29
3.4 CHAPTER SUMMARY........31
CHAPTER 4........32
4 Parallel Association Rule Mining Algorithms........32
4.1 Overview........32
4.2 Association rule mining on Apache Spark........32
4.3 PFP algorithm in Spark........33
4.3.1 PFP based on RDD........33

4.3.2 PFP based on Dataframe 33
4.4 Types of data that can be used in association rule mining algorithm 34
Text dataset 34
Tabular dataset 34
4.5 PFP algorithm step by step 34
4.5.1 Sharding 34
4.5.2 MapReduce 34
4.5.3 Grouping stage 35
4.5.4 Parallel FP-growth with MapReduce stages 35
4.5.5 Aggregation 36
4.6 PFP example 36
4.7 Chapter summary 39
CHAPTER 5 40
5 Semantic Knowledge Extraction in Big Data 40
5.1 Knowledge hierarchy in the context of IoT 40
5.1.1 Collect raw data 41
5.1.2 Semantic IoT data 41
5.1.3 Extract knowledge from semantic data 43
5.1.4 Actionable intelligence 44
5.2 Advantages of using association rule mining on semantic data 44
5.2.1 Improving RDF graph 44
5.2.2 Mining RDF graph and extracting association rules 45
5.3 Applying association rule mining on semantic data 46
5.4 Apache Spark as cloud to extract knowledge 47
5.5 Chapter summary 47

CHAPTER 6 48
6 Implementation 48
6.1 Implementation of FP-growth on semantic data using Apache Spark 48
6.2 Serialization of RDF graph into CSV file 48
6.3 Types of RDF data formats used 49
6.4 Combine values that occur at same time 54
6.5 Association rules mining steps 56
6.5.1 Feature selection 56
6.5.2 Preprocessing data 56
6.5.3 Transformation 57
6.5.4 Evaluation and interpretation 57
6.6 SAG algorithm by using SQL queries 59
6.6.1 SAG algorithm Example 59
6.6.2 Advantages of SAG algorithm 65
6.6.3 Disadvantages of SAG algorithm 65
6.7 CHAPTER SUMMARY 66
7 Evaluation 67
7.1 Overview of test datasets 67
7.2 NYC OpenData dataset 68
7.2.1 Public Pay Telephone Locations dataset 69
7.2.2 Queries 71
7.2.3 Benefits of using ARM on Public Pay Telephone Location dataset 76
7.2.4 Traffic Volume Count Dataset 76
7.2.5 Queries 80
7.2.6 Advantages of using ARM on traffic volume count dataset 87

7.2.7 Metal Content of Consumer Products Dataset 87

7.2.8 Queries 89

7.3 CityPulse dataset 90

7.3.1 Cultural Events Dataset 91

7.3.2 Road Traffic Dataset 92

7.3.3 Pollution Measurements for the City of Brasov in Romania 95

7.3.4 Advantage of using ARM on Road Traffic & pollution dataset 97

7.4 Kaggle dataset 97

7.4.1 Smart meters in Landon 97

7.4.2 Pima Indians Diabetes Database 101

7.4.3 Frequent two-factor set for Pima dataset by using SAG algorithm 106

7.5 The execution time for PFP algorithm on cluster of nodes 107

7.6 Chapter summary 108

CHPATER 8 110

8 Conclusion and Future work 110

8.1 Conclusion 110

8.2 Future work 112

References 113

LIST OF TABLES

Table 2-1 Transaction dataset........12
Table 2-2 Calculated frequency and priority........17
Table 2-3 Sorted items in transactions according to their priorities........17
Table 2-4 Conditional pattern base and conditional FP-tree........18
Table 3-1 Semantic diagnosis dataset........25
Table 3-2 sematic data for students........29
Table 3-3 candidate triples........30
Table 3-4 two itemset........30
Table 4-1 Style of dataset in ARM........33
Table 4-2 F-list........35
Table 5-1 Six configurations of context and target........45
Table 6-1 Store triples in csv file........50
Table 6-2 Triples (s, p, o)........60
Table 6-3 Result1 of SAG algorithm (one-factor set)........61
Table 6-4 Result of SAG algorithm (one factor set (support count for predicates))........62
Table 6-5 the result of **code 6-7**........63
Table 6-6 the result of **code 6-8**........64
Table 6-7 two-factor set........65
Table 6-8 The dataset after transformation step........66
Table 7-1 datasets........68
Table 7-2 ARM for PPTs........70
Table 7-3 Sub-model1 for PTTs........72
Table 7-4 Sub-model2 for PPTs ARM........73
Table 7-5 Sub-model for PTTs ARM........74
Table 7-6 Sub-model for PPTs ARM........75
Table 7-7 ARM for Traffic Volume Count Dataset........78

Table 7-8 Convert continuous numbers to categories 79
Table 7-9 ARM for Traffic volume when direction NB 80
Table 7-10 Sub-model for Traffic ARM when antecedent is Direction NB 81
Table 7-11 Trips in NB direction 82
Table 7-12 the trips in SB direction 83
Table 7-13 ARM for Traffic volume when direction is SB 84
Table 7-14 the trips in EB direction 85
Table 7-15 ARM when direction is EB 85
Table 7-16 The trips in WB direction 86
Table 7-17 ARM when direction is WB....................86
Table 7-18 ARM for metal content 88
Table 7-19 ARM for metal content when antecedent contains lead metal....................89
Table 7-20 ARM for products that made in China AND Lebanon 90
Table 7-21 ARM for Culture Events Dataset 92
Table 7-22 values that occur in the same hour for Road Traffic Dataset....................94
Table 7-23 Group number of vehicles 95
Table 7-24 ARM for Pollution Dataset 96
Table 7-25 Sample of smart meters dataset....................98
Table 7-26 RDF graph for weather dataset 99
Table 7-27 ARM for weather dataset 99
Table 7-28 classifier ARM for weather dataset....................100
Table 7-29 Convert dataset to triples 103
Table 7-30 ARM for diabetes dataset....................104
Table 7-31 ARM as classifier for diabetes dataset 105
Table 7-32 Frequent two factor set by SAG algorithm 106

LIST OF FIGURES

Figure 1-1 Semantic data example 4
Figure 1-2 MapReduce framework [https://www.tutorialspoint.com/map_reduce/map_reduce_introduction.htm] 5
Figure 1-3 (In memory execution) (Zecevic & Bonaci, 2016) 6
Figure 1-4 Python code showing the RDD methods flatMap, map,and reduceByKey (Word count example)..6
Figure 1-5 Wearable knowledge as a service in healthcare (generic conceptual semantic big data architecture)..8
Figure 2-1 1-itemset generation 12
Figure 2-2 2-itemset generation 13
Figure 2-3 Join 2-itemsets to generate 3-itemsets..13
Figure 2-4 Apriori property...14
Figure 2-5 Project FP-tree...18
Figure 2-6 Vertical dataset..20
Figure 2-7 Generate 2-itemset in Eclat algorithm..21
Figure 4-1 Sharding & MapReduce 37
Figure 4-2 Grouping & second MapReduce 38
Figure 4-3 Aggregation stage...38
Figure 5-1 Knowledge hierarchy in IoT..41
Figure 6-1 RDF/XML NYC dataset example 50
Figure 6-2 TURTLE data sample from CityPulse Dataset 52
Figure 6-3 Bnode Example 53
Figure 6-4 RDF graph data example from CityPulse dataset 54
Figure 6-5 Proposed methodology 58
Figure 7-1 RDF/XML PPTs dataset..69
Figure 7-2 Sample of RDF/XML Traffic Volume Count Dataset 77

Figure 7-3 RDF/XML Metal Content dataset 88
Figure 7-4 TURTLE Culture Event Dataset 91
Figure 7-5 Bnode as subject...98
Figure 7-6 Connect wearable devices to cloud 102
Figure 7-7 Performance of PFP algorithm on different size dataset 108

LIST OF CODES

Code 3-1 Calculate support count for target value 27
Code 3-2 Calculate support count for other predicates 27
Code 3-3 Generate 2-factorset 28
Code 6-1 Restore Bnode into CSV file..56
Code 6-2 Calculate the support count for target predicate 60
Code 6-3 Calculate the support count for target predicate 60
Code 6-4 Calculate the support count for non-target predicate......................................61
Code 6-5 Calculate the support count for non-target predicate......................................62
Code 6-6 Two-factor set 62
Code 6-7 First step to generate two-factor set 63
Code 6-8 Second step to generate two-factor set..63
Code 6-9 User defined function (conditional intersection) 64
Code 6-10 Cross join 64
Code 7-1 Extract I1 AND I2 from ARM model...71
Code 7-2 Extract I1 OR I2 from ARM model..74
Code 7-3 Filter I1 OR I2 from antecedent..75
Code 7-4 Convert continuous numeric data to categories 79

CHAPTER 1

1 Introduction

1.1 Overview

Association rule mining has long been a question of great interest in a wide range of fields such as the tourist route intelligent recommendation system (Chen & Zhou, 2015), Web Behavior (Lilleberg, 2015), and extracting rules from social media (Gole & Tidke, 2015). However, in the field of IoT (Internet of Things) applications, where all things have sensors and they are connected to the Internet, the generated raw data is too large and unlabeled. This has led researchers to add semantic annotations to represent IoT data (Barnaghi, Wang, Henson, & Taylor, 2012) such that sematic data or RDF graphs mean all data is represented as triples of subject, predicate, and object items. Subject and predicate values should be URIs while object may or may not be a URI. This representation can add rich information to IoT data because sematic data describes both the data and metadata for each sensor, including types of sensors, date and time of readings, and many other information values. However, with this increasing semantic data, mining RDF graphs has become an important issue in order to extract knowledge from huge volumes of semantic records of triples. The problem is that RDF graphs combine heterogeneous data from many domains, and subsequently mining a RDF graph of data becomes a critical challenge. Some researchers have been interested in classifying semantic data while others have focused on clustering (Onal, Sezer, Ozbayoglu, & Dogdu, 2017) and recently there has been a trend to extract association rules from triples (Barati, Bai, & Liu, 2017; Ramezani, Saraee, & Nematbakhsh, 2014). Both of these studies have used (predicate and object) together as one item, called a common behavior pattern. On the other hand, yet researchers have used SPARQL queries to mine RDF graphs (Tsay, Sukumar, & Roberts, 2015).

Most studies have only focused on mining semantic data in a sequential mode. However, in the field of IoT, the data is growing daily. Thus, handling large volumes of data in a sequential fashion may cause bottleneck problems especially in association rule mining algorithms because those algorithms are iterative algorithms and they require rescanning datasets frequently, and consequently the performance of such algorithms will be low. Therefore, scalable algorithms are required, or more clearly parallel association rule mining algorithms in a suitable framework are required to handle large volumes of data.

In this thesis, we studied parallel association rule mining algorithms. We tested the popular FP-growth algorithm on a number of datasets of varying sizes using a parallel implementation of the algorithm and reported on the performance of the algorithm.

1.2 Internet of Things and Big Data

The Internet of Things (IoT) is a connection between objects (physical items, humans, classes, plants, devices, and everything else) through a common network, usually a wireless network. Every item in the IoT has a unique identifier and every one of them works as an intelligent objects (Tsai, Lai, Chiang, & Yang, 2014).

IoT devices are connected to a network in two ways (Rozik, Tolba, & El-Dosuky, 2016):

- Directly connected in devices that have an operating system (e.g., smart phone).
- Indirectly connected in devices that do not have an operating system (e.g., Arduino).

IoT devices are increasing in number daily, in 2020 there will be more than 50 billion devices connected to the Internet (Rathore, Ahmad, Paul, & Rho, 2016). It has become necessary to handle this large volume of data in a scalable manner. In fact, we need hardware to store this huge amount of IoT data, including cloud storage, cores, memory, and distributed technologies. Moreover, we need software, such as scalable data mining algorithms to extract knowledge from IoT big data and enable sensors to take intelligent actions.

IoT data is usually unlabeled data and it does not give us a full description about the type of sensors, dates, locations. However, such information is very important in IoT applications. The RDF graph model is a way to represent data on the web with a specific schema (subject-predicate-object triples). This technique is added to IoT applications to make data understandable by machines. Moreover, an RDF graph can describe data and metadata for IoT data in the same graph.

1.2.1 The Semantic Web

"The Semantic Web is driven by the World Wide Web Consortium (W3C). It builds on W3C's Resource Description Framework (RDF), and is usually designed with syntaxes that use Uniform Resource Identifiers (URIs) to represent data".[1]

In RDF graph model, the data is represented as triples of subject, predicate, and object items. The Subject and predicate must be a URI while the object may be a URI or literal. When object is a URI, it may be subject to other triple, however RDF graph is directed and labeled graph, as a result all semantic data is understandable by the computer, moreover the semantic technique enables human to extract complex data by using SPARQL queries from the semantic web data.

---

[1] https://www.techopedia.com/definition/27961/semantic-web

Example:

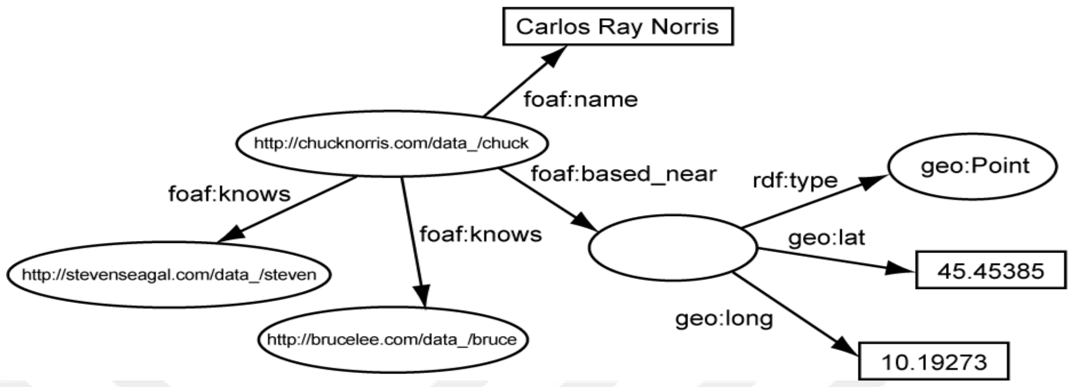


Figure 1-1 Semantic data example

From the graph above, we notice that the data is represented as a graph of triples. Each subject works as a primary key in the relational database, and the predicate describes the relation between the subject and object while the object is a value, literal, or a URI.

There are many advantages of using the semantic technique in IoT applications, such as interoperability, SPARQL queries, sharing information and many other advantages as explained by (Barnaghi et al., 2012).

In order to extract knowledge from a large semantic IoT dataset, we use machine learning algorithms. However, most machine learning algorithms are costly and they produce many intermediate results. When we implement machine learning algorithms on large volumes of data in one machine, bottleneck problem occurs. Therefore, it becomes necessary to distribute data to many machines to handle large amount of IoT data.

1.2.2 MapReduce

MapReduce is a programing model based on parallel processing. This model is used to handle large volumes of data. The Programmer has to write his own map function that depends on a key and value pair; therefore, each problem is expressed as a key and value pair. The map function will generate intermediate results, while the reduce function will aggregate all values with the same key (Dean & Ghemawat, 2008).

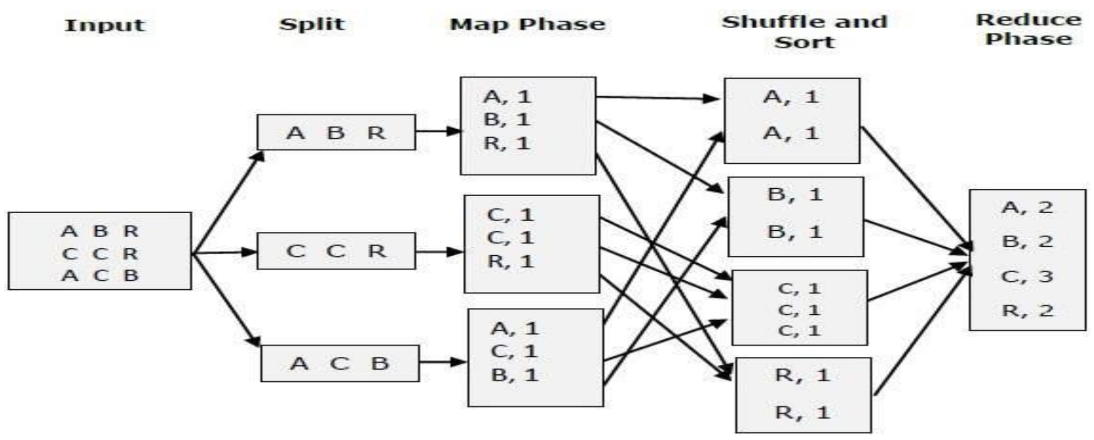


Figure 1-2 MapReduce framework [https://www.tutorialspoint.com/map_reduce/map_reduce_introduction.htm]

1.2.3 Apache Spark

Apache Spark is an open source framework that enables parallel processing of very large datasets. Spark is mainly designed to implement MapReduce paradigm of parallel processing. Moreover, it includes libraries for machine learning algorithms. An important advantage of Spark is its memory-based distributed data processing design (**figure1-3**), and it is suitable for iterative algorithms that require rescanning datasets many times. However, most machine learning algorithms are iterative and they spend a long time on I/O operations. In Apache Spark framework, we parallelize data on a cluster of nodes by creating RDD objects which is executed inside memory. This will optimize the performance of MapReduce algorithm. In **Figure 1-3** below, the dataset is stored in a cluster of nodes (RDD), and if we need to repeat scanning dataset, we can easily do so the data is stored in the memory and we do need to fetch the data from hard disk storage.

In Apache Spark, we parallelized the data by creating RDD (Resilient Distributed Datasets). This RDD is considered to be object **(Zaharia et al., 2012)** and this object has methods including map, reduce, filter, union, and other functions that can be applied to the dataset partitions in parallel mode. An RDD (Resilient Distributed Datasets) object in memory execution and it consists of distributed collection of data as shown in **figure1-3**.

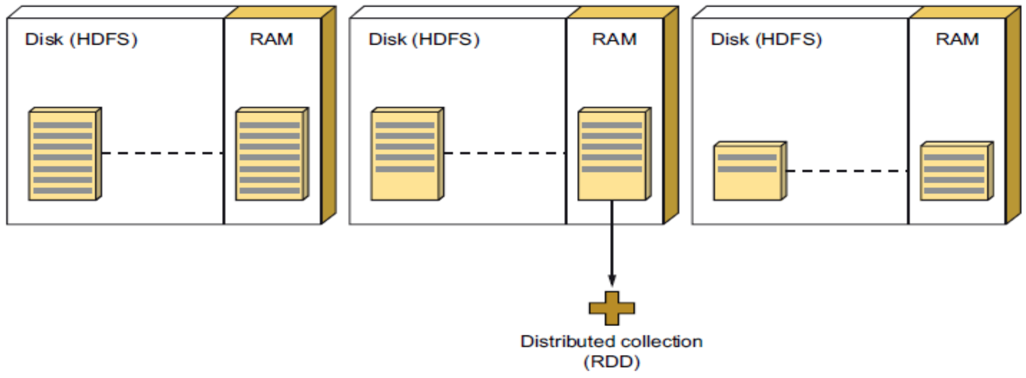
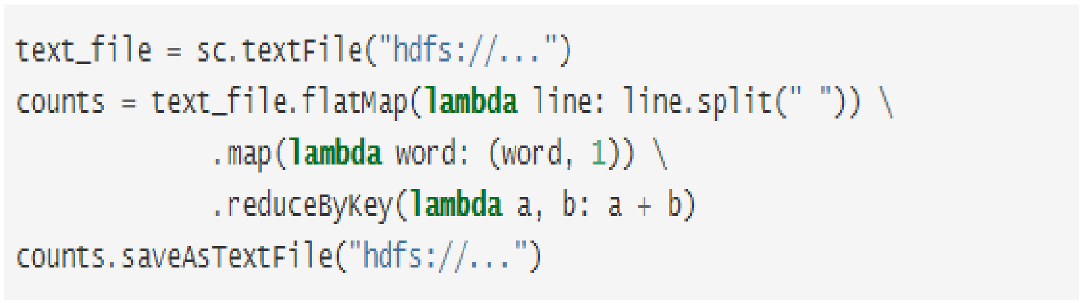


Figure 1-3 (In memory execution) (Zecevic & Bonaci, 2016)

Actually the RDD is partitioned in a cluster of nodes that can be handled in a parallel fashion[2]. For example, if we have a text file and we use the map function on our RDD, then each line will be mapped independently in parallel simultaneously with other lines. However, all of those functions or transformations are called lazy mode; because they do not return any intermediate result yet; Therefore, the main memory can fit the data and the computer does not need to store large intermediate results in hard disk storage. The following example in the **Figure1-4** explains a number of RDD functions.



```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
             .map(lambda word: (word, 1)) \
             .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

Figure 1-4 Python code showing the RDD methods flatMap, map,and reduceByKey (Word count example)

[2] https://spark.apache.org/docs/latest/rdd-programming-guide.html

1.2.4 Association Rule Mining (ARM) on IoT data

Association Rule Mining is one of the most important machine learning techniques which extracts frequent patterns and describes the correlation between items that appear together in a large dataset. An example can include any items are purchased together in the market basket analysis.

IoT sensors and devices are not intelligent by themselves. However, the data collected from these devices can be used to create and design intelligent systems, such as smart homes, smart grids, or smart cities, by applying machine learning techniques to an IoT dataset. One such method is “association rule mining” which is used to detect and learn frequently occurring patterns in IoT data.

Association rule mining has been used in many smart home systems the papers of (S. Li & Zhou, 2017; Yassine, Singh, & Alamri, 2017) applied association rule mining to extract frequent human activities in smart homes. The second paper used knowledge of association rules in healthcare systems to detect abnormal activity in patients. This type of mining is also used in (Kang, Ka, & Kim, 2012) to monitor elderly people. This application is called the Elderly Surveillance System.

In the field of semantic data, IoT, wearable devices, and data mining, an interesting study was proposed by (Mezghani, Exposito, Drira, Da Silveira, & Pruski, 2015) to connect all wearable devices for all patients, such as heart rate sensors, pressure sensors and glucose patch into one cloud in which the cloud would receive data and convert that data into semantic data. Then the machine learning algorithms and MapReduce would be applied to extract the state of the patient as either normal or if a disease is detected, another fact will be added to the knowledge base. This can help physicians to diagnose the state of patient. Moreover, physicians would be able to see all information for each patient at any time because the data is semantic and the metadata (data explaining data) would also be available.

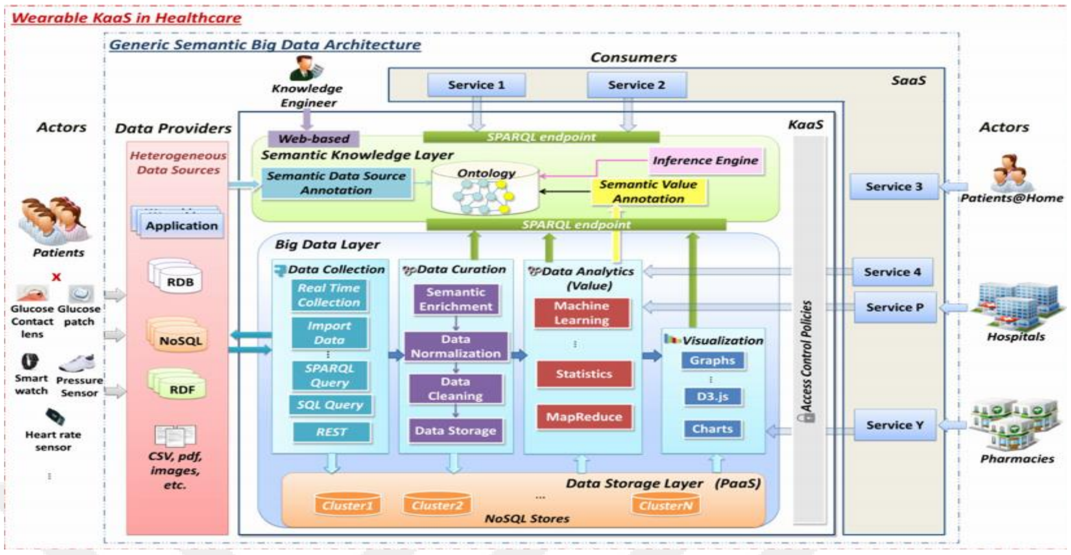


Figure 1-5 Wearable knowledge as a service in healthcare (generic conceptual semantic big data architecture)

1.3 Objective

The specific objective of this thesis is to parallelize the FP-growth algorithm on semantic IoT data and extract frequent patterns from semantic big IoT data. While (predicate and object data) is considered as one pattern, called the common behavioral pattern, ARM has been used to improve RDF graph (Abedjan & Naumann, 2013) by using six types of configurations. In this study we used one of those configurations, and in fact we mined the object when the context was the predicate. This type of mining was used to discover a range of continuous values and to summarize the RDF graph to avoid storage explosion.

Another important objective of this study is to extract specific interesting patterns using the ARM model. Therefore, after the ARM model is created, we use SQL queries in the PySpark[3] framework, which is the Python API for Apache Spark, to create sub-models of ARM. The results show that ARM can work as a classifier in a number of cases.

[3] PySpark: http://spark.apache.org/docs/2.2.0/api/python/pyspark.html

1.4 Organization of the Thesis

In Chapter2 we explain the state-of-the-art in association rule mining and its algorithms. Chapter3 covers the related work on association rule mining on semantic data. Then, in Chapter4 we describe the Parallel FP-growth (PFP) algorithm which is one of the parallel algorithms for association rule mining on distributed data collections. The algorithm was originally proposed by Wang et al (2008) and it is also implemented in Apache Spark framework. Chapter5 explains the knowledge hierarchy in IoT applications and it covers the purpose of ARM in this study. Chapter6 describes the methods used in this study in details while Chapter7 focuses on the evaluation of the parallel semantic association rule mining on a number of datasets that are used in this study and we report the performance and the results on each dataset. Finally in Chapter8, we conclude and point to future work in this area.

CHAPTER 2

2 Association Rule Mining

In this chapter, we first describe a popular machine learning method called Association Rule Mining (ARM). Then, we explain in detail two popular ARM algorithms, namely Apriori and FP-growth. And finally, we explain ARM algorithms applied on vertical datasets such as ECLAT (Equivalence Class Transformation) and DECLAT (Diffset Equivalence Class Transformation) and the difference between them.

2.1 Definition

Association Rule Mining (ARM)[4] is a popular rule-based machine learning method that is used to find relationships between variables in datasets. ARM takes the following form: **IF (Antecedent) THEN (Consequent)**. For example, in market basket analysis, we extract association rules between items that are bought together such as bread and milk. The rule will then be IF (bread) THEN (milk) with confidence C. This rule means if you buy bread, then you may buy milk with some confidence C. ARM describes potential rules between items with specific confidence and support.

[4] Association Rule Mining: https://en.wikipedia.org/wiki/Association_rule_learning

2.2 Apriori algorithm

Apriori is the most popular algorithm for ARM. It has been proposed by R. Agrawal and R. Srikant in 1994 (Pei & Kamber, 2011) to extract frequent patterns from market basket datasets.

This algorithm has been used to generate association rules and relationships between items that occur frequently in large datasets that contains large numbers of transactions.

Apriori algorithm is easy to run when the dataset is small, but when we have a large dataset, this algorithm will take a very long time to execute and also large memory to handle the execution.

Apriori algorithm depends on two main criteria:

- Minimum support S = P(A ∪ B), probability of transactions that contain A, B together in all database transactions. This is calculated as:

  $$\text{Support}(A \cup B) = \frac{Support\ Count(A \cup B)}{Number\ of\ transactions}$$

- Minimum confidence C=A | B, conditional probability of transactions that contain A also contain B. This is calculated as:

  $$\text{Confidence}(A \mid B) = \frac{Support\ Count(A \cup B\ )}{Support\ Count(A)}$$

All generated rules should satisfy the above thresholds criteria to pass the test.

2.2.1 Apriori algorithm example

Apriori algorithm works as in the following example:

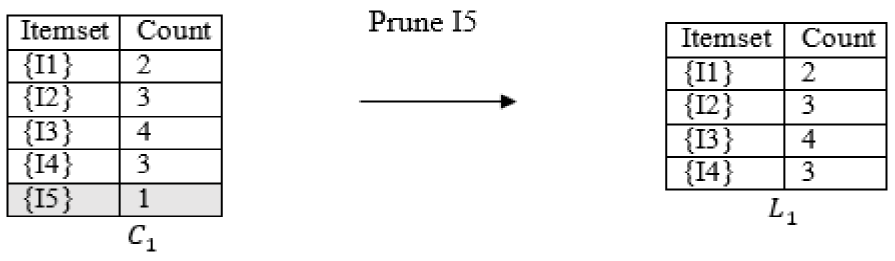
Assume we have 5 transactions as shown in **Table 2-1**. Each transaction has a unique transaction ID TID., This table consists of five transactions and the number of items in transactions is k=4 (I1, I2, I3, I4).

Table 2-1 Transaction dataset

TID	Transactions
1	I2, I5
2	I3, I4
3	I2, I3, I4
4	I1, I2, I3
5	I1, I3, I4

Assume that the minimum support is 40%, that means each element should occur at least two times because the support count = (40/100) x 5=2, and 5 refers to number of transactions. If the minimum confidence is 60%, then the minimum confidence count is (60/100) x 5=3.

First, we calculate support count for each item by scanning all transactions in the dataset and generate what is called candidate 1-itemset or C1, we prune all items that do not satisfy minimum support 2, after pruning step we get frequent 1-itemset L1 as follows:



Itemset	Count
{I1}	2
{I2}	3
{I3}	4
{I4}	3
{I5}	1

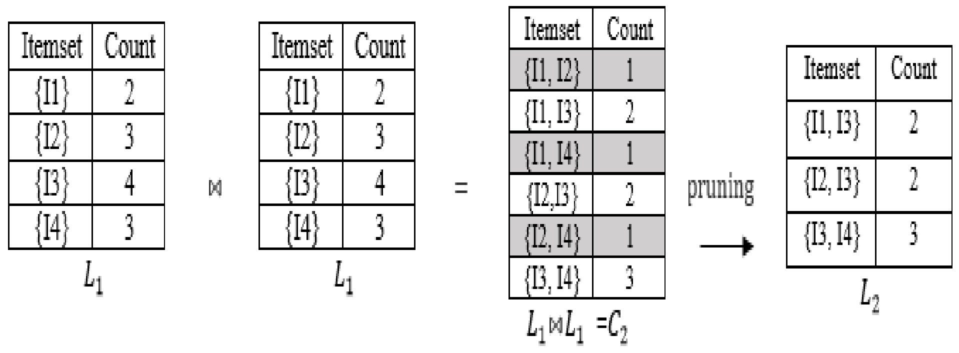
C_1

Itemset	Count
{I1}	2
{I2}	3
{I3}	4
{I4}	3

L_1

Figure 2-1 1-itemset generation

Second, we join L1 with itself to generate candidate 2-itemset C2. Apriori algorithm assumes that itemsets are stored in lexicographic order to ensure there are not duplicated items in similar itemsets. After candidate itemsets are generated, again we scan the dataset and calculate support count for each itemset in C2, if the support count of itemset < minimum support, we will prune this itemset. After pruning step, we get 2-itemset L2 as follows:



L_1

Itemset	Count
{I1}	2
{I2}	3
{I3}	4
{I4}	3

$\bowtie$

L_1

Itemset	Count
{I1}	2
{I2}	3
{I3}	4
{I4}	3

=

$L_1 \bowtie L_1 = C_2$

Itemset	Count
{I1, I2}	1
{I1, I3}	2
{I1, I4}	1
{I2,I3}	2
{I2, I4}	1
{I3, I4}	3

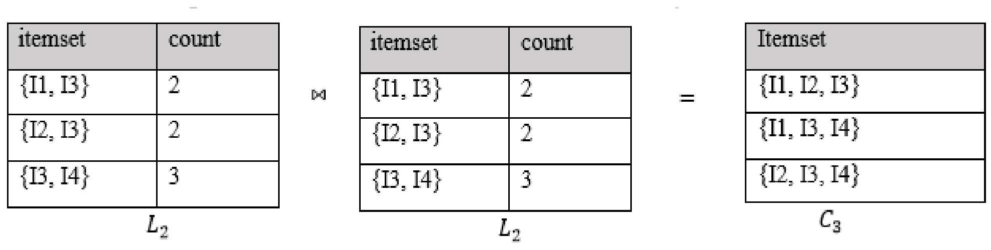
pruning $\longrightarrow$

L_2

Itemset	Count
{I1, I3}	2
{I2, I3}	2
{I3, I4}	3

Figure 2-2 2-itemset generation

Third, join L2 with itself to generate candidate 3-itemset C3 similarly:



L_2

itemset	count
{I1, I3}	2
{I2, I3}	2
{I3, I4}	3

$\bowtie$

L_2

itemset	count
{I1, I3}	2
{I2, I3}	2
{I3, I4}	3

=

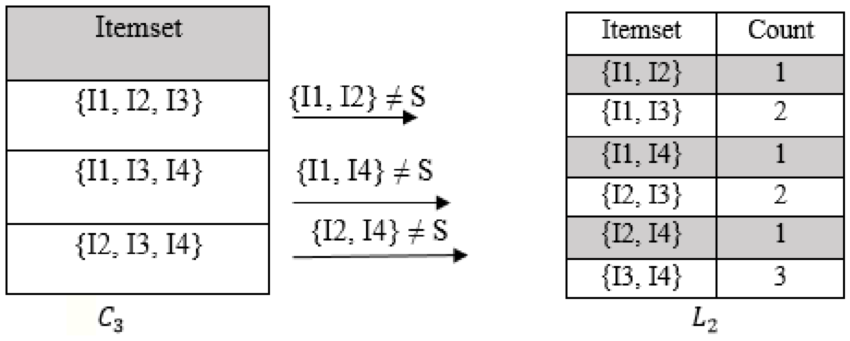
C_3

Itemset
{I1, I2, I3}
{I1, I3, I4}
{I2, I3, I4}

Figure 2-3 Join 2-itemsets to generate 3-itemsets

Depending on Apriori property (if sub-set of itemset is not frequent, the whole itemset will not be frequent also) (Pei & Kamber, 2011), in C2, the itemset [I1, I2] ⊂ [I1, I2, I3], but [I1, I2] is not frequent, so the itemset [I1, I2, I3] is also infrequent and it will be pruned.

-



Itemset
{I1, I2, I3}
{I1, I3, I4}
{I2, I3, I4}

Itemset	Count
{I1, I2}	1
{I1, I3}	2
{I1, I4}	1
{I2, I3}	2
{I2, I4}	1
{I3, I4}	3

Figure 2-4 Apriori property

Since all itemsets consist of non-frequent sub-itemsets for example [I1, I2, I3] consists of [I1, I2], [I1, I3] and [I2, I3] however support count for [I1, I2] less than minimum support so [I1, I2, I3] will be pruned, because sub-itemset of it is not frequent according to Apriori property "*if itemset a is not frequent then a ∪b also is not frequent*".

Because all sub-itemsets of C3 are not frequent, so L3= Ø, here we are stopping algorithm. Last step is generating association rules for frequent 2-itemsets L2, each generated rule in L2 should satisfy to minimum confidence threshold C = 60%, and minimum support S = 40% , we will test all itemsets in L2, to check whether they can be association rule or no.

[I1, I3] association rules

- I1→I3, C = 2/2 = 100%, S = 2/5 = 40%
- I3→I1, C=2/4 = 50%, S = 2/5 = 40%

[I2, I3] association rules

- I2→I3, C=2/3 = 66%, S=2/5 = 40%
- I3→I2, C=2/4 = 50%, S=2/5 = 40%

[I3, I4] association rules

- I3→I4, C=3/4=75%, S= 3/5=60%
- I4→I3, C=3/3=100%, S=3/5=60%

All rules with confidence less than 60% or support less than 40% will be prunes, as a result the rules that pass two thresholds are I1→I3, I2→I3, I3→I4 and I4→I3

2.2.2 Limitations of Apriori algorithm

Generating candidate itemsets will produce large itemsets that require large memory (costly), especially when the dataset is large and the minimum support is low.

Apriori is an iterative algorithm, that means it repeats scanning data set to calculate support count for each generation of candidate itemsets, and as a result the performance in time complexity will be poor, the process can take a long time and possibly fail..

2.2.3 Improving the efficiency of Apriori algorithm

Many algorithms have been proposed to improve Apriori algorithm (Kothari & Patel, 2015; Mohammed Javeed Zaki, 2000; Mohammed J Zaki & Gouda, 2003). All of these algorithms aim to reduce time complexity and memory usage as a result of repeat scanning database and iterations. We will explain some of these methods in related works:

- Hashing technique (Park, Chen, & Yu, 1995):The hashing technique is an efficient storage method that has been used to facilitate insert, retrieve, and delete items from database. Here we can use hash table as follows:
    - Scan all transactions in the dataset and calculate support-count of each item and remove items that do not satisfy the minimum support, and generate C1 (candidate 1- item set).
    - By using C1 we generate C2 and store it in a hash table by using hash function h(x, y)=((order of x)*10 +(order of y)) % n (Park et al., 1995),while n is the size of hash table.
    - This technique has been used to reduce the size of candidate itemsets.

  - Many Apriori algorithms based on this technique have proposed to reduce the time and size of candidate 2-itemset. Incremental association rule miming algorithms have been applied on dynamic databases (Kothari & Patel, 2015). This algorithm uses hash table to store candidate 2-itemsets, because the database is updating and the pervious association rules will be invalid, and rescanning the database will take a long time, by hashing technique the size of candidate 2-itemsets are reduced that lead to reduced execution time.
- Another technique is reducing the number of transactions in the database, if the transaction does not contain any frequent items, they will be removed from database.
- Partitioning database so that each partition can fit in main memory such as MapReduce framework.

2.3 FP-growth algorithm

It is an efficient and scalable algorithm to extract rules from a dataset without generating candidate itemsets. It consists of two phases:

- Build an FP-tree, where each branch of tree presents transactions in dataset.
- Extract frequent patterns directly from tree.

FP-algorithm is better than Apriori algorithm because it reduces the cost of candidate itemset generation and the cost of rescanning the dataset. We can explain this algorithm with the following example:

Assume the dataset in Table 2-2 with 5 transactions. The first step of FP-growth is scanning the dataset and calculating the support count for each item in the dataset. This step is similar to the first step in Apriori algorithm. Because I5 has the lowest minimum support, it will be pruned, then we sort items based on the priority of support count for items.

In Table 2-2, the item I3 has the highest priority because it is repeated 4 times, so I3 takes number 1, then both of I2 and I4 are repeated 3 times, we give I2 the priority 2 and I4 the priority 3, while I1 takes the priority 4.

Table 2-2 Calculated frequency and priority

Item	Frequency	Priority
I1	2	4
I2	3	2
I3	4	1
I4	3	3
I5	1	

In the second step, items in transactions are ordered according to their priorities (Table 2-3).

Table 2-3 Sorted items in transactions according to their priorities

TID	Transactions	Ordered items
1	I2, I5	I2
2	I3, I4	I3, I4
3	I2, I3, I4	I3, I2, I4
4	I1, I2, I3	I3, I2, I1
5	I1, I3, I4	I3, I4, I1

In the third step, the root of FP-tree is created as a null node so this root is considered as the parent node for all nodes in the FP-tree (**Figure 2-5**). After that, we match each transaction in the dataset as a branch of FP-tree, and count the transactions that has the same ordered items from the root node.

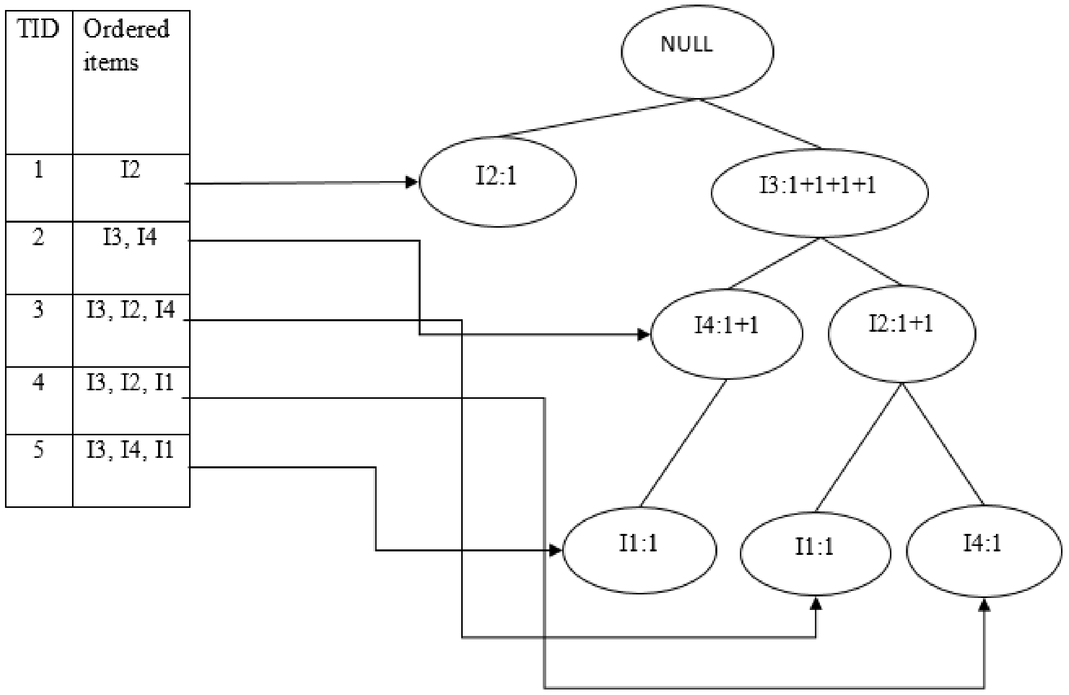


Figure 2-5 Project FP-tree

In the fourth step, mining FP-tree by generating two main things, conditional pattern base and conditional FP-tree for each item in the dataset starting with the item that has the lowest priority as following:

Table 2-4 Conditional pattern base and conditional FP-tree

item	Coditional pattern base	Conditional FP-tree	Frquent patterns generated
I1	[I3,I4]:1, [I3, I2]:1	[I3:2]	[I3, I1]:2
I4	[[I3, I2]:1, [I3]:1]	[I3:2]	[I3, I4]:2
I2	[I3]:2	[I3:2]	[I3, I2]:2

In conditional pattern we start from items that have lower priority as suffix of branch. In this example, I1 is the lowest priority, I1 is constructed by two paths [I3, I4]:1 and [I3, I2]:1, but I2 and I4 have count 1 less than support count, therefore I2 and I4 are excluded from conditional FP-tree therefore I1 can generate one frequent pattern [I3, I1]. Similarly the item I4 has two conditional pattern base [I3, I2]:1 and [I3]:1. Here, I2 is repeated one time with I4, so I2 has a count less than the support count, therefore conditional FP-tree consists of [I3] only and frequent pattern generated is [I3, I4]:2 as the support count. Finally, the conditional pattern base of I2 is [I3]:2 and the conditional FP-tree is also [I3]:2, and the frequent pattern generated is [I3, I2]:2.

Finally, we generate association rules from the frequent patterns. For example from [I3, I4], we can generate two rules [I3, I4] and [I4, I3] by calculating the confidence for each rule. This step is similar to the last step in Apriori algorithm.

2.3.1 Advantages of FP-growth algorithm

- FP-growth algorithm requires only two scans of a dataset. This reduces the execution time and make this algorithm simpler and faster than Apriori algorithm.
- FP-growth can generate frequent patterns without candidate generating, thus reduce the cost of required memory to store many candidates itemsets.
- This algorithm can directly extract frequent patterns after constructing an FP-tree and also find conditional pattern base for each item.

2.3.2 Disadvantages of FP-growth algorithm

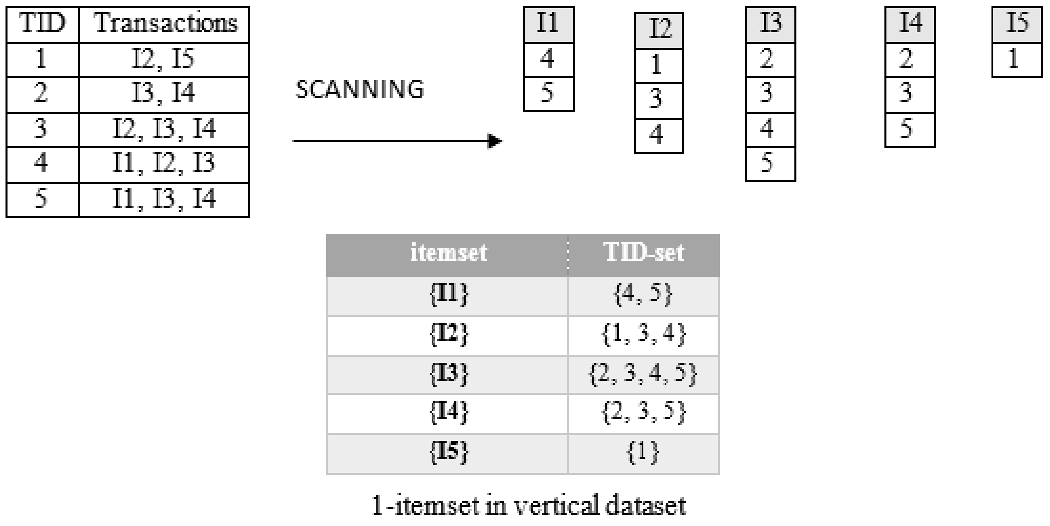
- FP-tree consists of branches, nodes, pointers, and counters for each item. Therefore, when a dataset is too big, the FP-growth algorithm will generate a large tree and require larger memory space.
- In transaction dataset, usually many transactions share some items and share nodes in FP-tree, but if many transactions have unique prefix, the generated tree will be larger than the original dataset.
- A large tree may not fit in main memory and storing tree in disk space will take long time causing poor execution performance.

2.4 Mining frequent patterns by using vertical data format

2.4.1 ECLAT algorithm

All pervious algorithms work on horizontal datasets in which each transaction contains a number of items. Vertical dataset processing for frequent patter mining is proposed by (Mohammed Javeed Zaki, 2000). The algorithm is called Eclat (equivalence class transformation). The idea behind this algorithm is using all items in dataset together as root, and then classifying the root into independent classes and applying intersection operation on classes that share the same prefix. This algorithm could outperform Apriori algorithm and it got efficient results. This algorithm works as follows:

- Convert horizontal dataset into vertical dataset by scanning dataset once while rows represented as item numbers in dataset and columns represented as TID-set or number of transactions corresponding with each item, this step is the most important step to reduce the number of scanning of dataset first and to minimize the magnitude of dataset, so the dataset can be handled in memory. For example, from table [1] we can convert dataset to vertical format as shown in Figure 2-6.



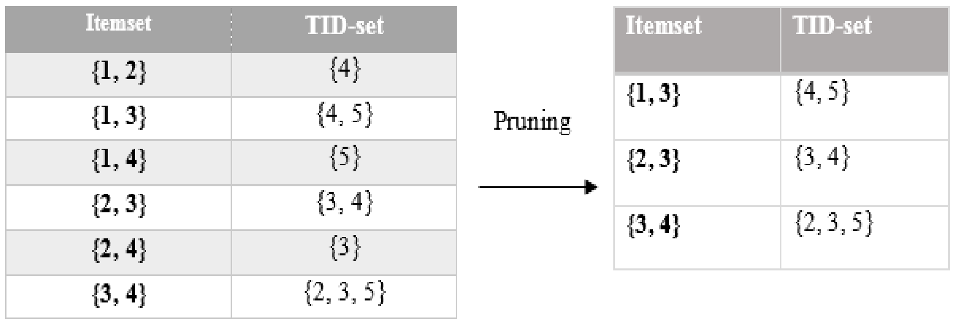
TID	Transactions
1	I2, I5
2	I3, I4
3	I2, I3, I4
4	I1, I2, I3
5	I1, I3, I4

I1	I2	I3	I4	I5
4	1	2	2	1
5	3	3	3	
	4	4	5	
		5		

itemset	TID-set
{I1}	{4, 5}
{I2}	{1, 3, 4}
{I3}	{2, 3, 4, 5}
{I4}	{2, 3, 5}
{I5}	{1}

Figure 2-6 Vertical dataset

- The support count of each itemset is the number of transactions in TID-itemset, so we will prune [I5] itemset because the support count of it less than the minimum support.
- We can get 2-itemset in vertical dataset by intersection operation between 1-itemsets, then pruning itemsets that have minimum support less than the minimum support threshold as follows (**Figure 2-7**):



Itemset	TID-set
{1, 2}	{4}
{1, 3}	{4, 5}
{1, 4}	{5}
{2, 3}	{3, 4}
{2, 4}	{3}
{3, 4}	{2, 3, 5}

Pruning

Itemset	TID-set
{1, 3}	{4, 5}
{2, 3}	{3, 4}
{3, 4}	{2, 3, 5}

Figure 2-7 Generate 2-itemset in Eclat algorithm

- Depending on apriori property all itemsets that contain non-frequent itemsets, they will be also non-frequent, we see all [I1, I2], [I1, I4], [I2, I4] are non-frequent itemsets. However, all 3-itemsets contain those itemsets, so we do not need to generate 3-itemsets and make intersection operation because 3-itemset=Ø and here the algorithm is stopped.

2.4.1.1 Advantages of ECLAT algorithm (Mohammed Javeed Zaki, 2000)

- Reduce scanning of dataset (only one time) thus reducing the cost of I/O operations,
- Intersection operation can be done by SQL queries,
- Minimize database size, especially in sparse datasets,
- It is more scalable and faster than Apriori algorithm.

2.4.1.2 Disadvantages of ECLAT algorithm

- Intersection operation can take long time,
- Intermediate results may be large, so those results need to be written disk.
- In large and dense datasets, the generated TID-set dataset may be too big.

2.4.2 DECLAT algorithm (Mohammed J Zaki & Gouda, 2003)

DECLAT algorithm is proposed in 2003. This algorithm is developed to avoid Eclat constrains, it is working as ECLAT algorithm, but it uses differences between classes instead of intersection operations. This technique could improve the performance of ECLAT algorithm in efficiency especially in dense datasets.

2.4.3 Comparison between ECLAT and DECLAT algorithms:

- ECLAT algorithm can work well in sparse datasets; because the generated dataset will be small.
- DECLAT algorithm can work well in dense datasets; because the dataset will be minimized in size.
- DECLAT algorithm is developed to "make ECLAT algorithm more scalable" (Mohammed J Zaki & Gouda, 2003), but it depends on the type of dataset, so that we can choose one of them.
- Experiments show that both ECLAT and DECLAT algorithm can work together for effective results. When dataset is sparse, we use ECLAT algorithm, then switch to DECLAT algorithm, but when dataset is dense, we use DECLAT, and then switch to ECLAT algorithm.

2.5 CHAPTER SUMMARY

In this chapter, we have explained the most important and common ARM algorithms. Apriori is the most popular algorithm but it is costly and it is not appropriate for large datasets. We also discussed FP-growth algorithm which is better than Apriori because it does not repeat scanning the dataset more than two times. Then, we explained ECLAT and DECLAT algorithms that require projecting the dataset into a vertical format. Also we discussed the advantages and disadvantages of each algorithm. Although all these algorithms can improve the performance of Apriori algorithms, they still suffer from efficiency problems when datasets are too big; Therefore, we need more scalable algorithms that can handle large dataset efficiently.

We noted that all of Apriori, FP-growth, ECLAT, and DECLAT algorithms can extract Association Rules from traditional transaction datasets, but they cannot handle semantic datasets immediately.

Due to the limitation of capabilities of these algorithms to handle large datasets and extract association rules from semantic data, we will discuss other algorithms that can handle semantic data in the next chapter.

CHAPTER 3

3 Semantic Association Rule Mining

Association rule mining (ARM) is a common data mining technique, which has considerable impact on extracting knowledge from big datasets. This technique has been used by many researchers in many fields, such as positive association rule mining in tourist route intelligent recommendation systems (Chen & Zhou, 2015), web behavior extraction (Lilleberg, 2015), extracting rules from social media (Gole & Tidke, 2015), or negative association rules to extract infrequent patterns in intrusion detection systems (Kong, Jong, & Ryang, 2016). Applying association rule mining on RDF (semantic) data, where data is represented as triples (subject-predicate-object) is a challenging issue due to the heterogeneity of data in the Web first, and then the increasing volume of semantic RDF data on the Web secondly. Besides, in RDF graphs there is no exact definition of transactions (Ramezani et al., 2014), all data is stored as triples (subject, predicate, and object). In this chapter we present the literature review on Association Rule Mining on semantic datasets. We explain three recent ARM algorithms on RDF graphs. These are SWApriori (Ramezani et al., 2014), SAG (Tsay et al., 2015), and SWARM (Barati et al., 2017).

In recent years, there has been an increasing interest in mining RDF graphs, however extracting association rules from RDF data is one of the most important techniques in mining RDF graphs. The researchers have introduced different algorithms for ARM on RDF graphs. We will introduce the recent works on ARM for semantic data in the following sections.

3.1 SWARM algorithm

SWARM algorithm (Semantic Web Association Rule Mining) (Barati et al., 2017) is the most recent work of ARM for semantic data. This algorithm has worked in both schema level and instance level. For example, if the predicate is `rdf:type`, the algorithm will not ignore it, but it will use it to define the final rule. Assume we have the following table of triples (**Table 3-1**).

Table 3-1 Semantic diagnosis dataset

subject	predicate	Object
Ali	Take	Panadol
Ali	Type	Renal failure patients
Ali	Suffer	Stomach
Mona	Type	Renal failure patients
Mona	Type	Woman
Moan	Suffer	Headache
Mona	Take	Panadol
Jon	Take	Pensilen
Jon	Type	Renal failure patients
Jon	Suffer	Headache
Jack	Type	Renal failure patients
Jack	Suffer	Headache
Sara	Take	Panadol
Sara	Suffer	Stomach
Yazen	Take	Panadol
Yazen	Suffer	Headache

The first step of this algorithm is to extract sematic items (SI). In the above table each (predicate & object) is called "patient behavior". For example, `(take, Panadol)` is considered as the same behavior shown by the subjects `Ali, Mona, Sara,` and `Yazen`. Therefore, all subjects with the same behavior will be added together to construct the first semantic item, named semantic item SI1 as `[Ali, Mona, Sara, Yazen]: (take, Panadol)`. Similarly, SI2 = `[Mona, Jon, Yazen]: (suffer, Headache)` can be constructed.

The second step is to generate common behavior items (CBS). Here, the algorithm will receive all SI from the last step, and calculate the similarity degree (SD) for each SI. For example, SD for SI2 is 3, because it contains three subjects. By assuming the similarity threshold is equal 2.

```
CBS1 [[Mona, Yazen, Ali, Sara]: (take, Panadol)

        [Mona, Yazen, Jon]: (suffer, Headache)]]

CBS1 [Mona, Yazen]: [(take, Panadol), (suffer, Headache)]
```

The similarity degree SD for CBS1 is 2 because the number of subjects equal 2; therefore, we can see that SD for CBS1 is equal to similarity threshold, if SD less than similarity threshold, the CBS1 will be ignored.

The third step is to generate semantic association rules by taking a union of CBS items. therefore R1 will be `R1= [Mona, Yazen]:(take, Panadol) → (suffer, Headache)`. It is also possible to change the order of items. The last step of this algorithm is to define the schema level for each rule, which is not important for us in this study.

Although SWARM algorithm seems easy, the pseudo code of this algorithm contains many loops, which clearly takes a long time to execute, especially for big data.

3.2 SAG algorithm

This algorithm, proposed by (Tsay et al., 2015), enables us to mine RDF graph directly by using SPARQL language. We do not need to restore triple data in tabular format, but just apply a series of SPARQL queries on the data and generate what is called one factor set and two factor set, and so on. This algorithm enables users to determine target predicates. For example, from the above table (Table 3-1) if we consider (`suffer`) as the target value and we need to mine objects that are related to this target value, by setting the minimum support to 1, we follow these steps:

First, scan the table to calculate the support for target predicate value, where the support of a predicate is the number of subjects that are related to this predicate and object. In this case we will find following result:

Result11= [(suffer, headache):4, (suffer, stomach):2]

The SPARQL query to generate these candidates, according to SAG algorithm is:

```
SELECT ?p ?o (COUNT(*) AS ?sup)
WHERE {
  ?s ?p ?o.
  FILTER (regex(str(?p), ‘suffer’, ‘i’)).
}
GROUP BY ?p ?o
HAVING (?sup>=2).
```

Code 3-1 Calculate support count for target value

Second, calculate the support count for the other predicates where the support is the number of subjects for each pair of (predicate + object), except the target predicate. The APARQL query for this frequent item mining is:

```
SELECT ?p ?o (COUNT(*) AS ?sup)
WHERE {?s ?p ?o.
        FILTER (!regex(str(?p), ‘suffer’, ‘i’)).}
 GROUP BY ?p ?o
HAVING (?sup>=2).
```

Code 3-2 Calculate support count for other predicates

The result of the above query is:

Result12= [(type, renal failure patients):4, (type, woman):1, (take: Panadol):4]

All of result11 and result12 are called one factor set. In order to generate two factor set, we will make another query that combines the two above results and generate another factor set.

The second query to generate two factor set, actually this query joins the two above queries, the first part of this query is similar to the first query, while the second part as second

```
SELECT ?p1 ?o1 ?p2 ?o2 (COUNT(*) AS ?sup)
WHERE {?s ?p1 ?o1.
        FILTER (regex(str(?p1), ‘suffer’, ‘i’)).}
                ?s ?p2 ?o2
        FILTER (!regex(str(?p2), ‘suffer’, ‘i’)).}
GROUP BY ?p1 ?o1 ?p2 ?o2
HAVING (?sup>=2)
ORDER BY ?p2 ?o2
```

Code 3-3 Generate 2-factorset

query, therefore the third query repeats the same scanning and counting, but by taking the intersection between the two results as shown below.

Above query will generate two factor set `[[(type, woman), (suffer, headache)]:1, [(type, renal failure patient), (suffer, headache)]:3, [(take, Panadol), (suffer, headache)]:2, [(type, renal failure patient), (suffer, stomach)]:1, [(take, Panadol), (suffer, stomach):2`].

We will repeat the queries until no candidate items are generated.

Representing data as RDF triples and applying association rule mining as shown in the above examples can play an important role in extracting very useful information, and int this case a diagnosis for the patient conditions. This knowledge can be shared by the others in many hospital systems. Besides, this type of mining integrates between different domains, for example in the hospital system, there are many departments such as diagnosis, laboratories, and pharmacy departments. If we apply association rule mining for the same patient, we will be able to understand the whole information about the patient and extract frequent patterns for many patients, and consequently we can diagnose patient conditions or share this information with other doctors and hospitals, leading to may be identifying epidemics, or other public health problems.

3.3 SWApriori

SWApiori algorithm (Ramezani et al., 2014) is used to mine triples (subject, predicate, and object). This algorithm combines Apriori and ECLAT algorithms, and its implementation consists of the following steps:

- store triples in a table of three columns with subject, predicate, and object
- convert data to numbers
- define items as entity plus relation, where entity is object and relation is predicate
- the items that are repeated in many triples will be considered as one itemset,

For example, if we assume the triples in the table below:

Table 3-2 sematic data for students

subject	predicate	object
Omar	study	math
Laila	Supervised by	Ahmed
Ahmed	Supervised by	Erdogan
Osama	Supervised by	Erdogan
Mohanad	study	CENG
Ahmed	study	CENG
Mustafa	Supervised by	Erdogan
Mustafa	study	CENG

We notice that (`supervised by, Erdogan`) are repeated three times, we can calculate the minimum support for this item, by calculating the number of subjects for the same item (predicate, object), and therefore the first item set will be as follows:

Table 3-3 candidate triples

subject	predicate	object
Ahmed	Supervised by	Erdogan
Osama	Supervised by	Erdogan
Mohanad	study	CENG
Ahmed	study	CENG
Mustafa	Supervised by	Erdogan
Mustafa	study	CENG

All items (predicate, object) that have low minimum support are ignored such as (study, math) because it is repeated just one time where subject is Omar.

Third, we generate large item sets, in this example to generate two itemset we make an intersection between the subjects for the candidates triples.

Table 3-4 two itemset

Subject	predicate	object
Ahmed, Osama, Mustafa	Supervised by	Erdogan
Mohanad, Ahmad, Mustafa	study	CENG

The result of the intersection operation for the above two triples is (Ahmed, Mustafa). By assuming that the minimum support is 2, therefore we can join two triples together as two itemsets.

Therefore, the two item sets will be [(supervised by, Erdogan), (study, CENG)], if those triples represent data in Cankaya University, then we can say

[Most of students that study CENG in Cankaya university, their supervisor is Erdogan]

Consequently, SWApriori algorithm will generate 3 itemsets and 4 itemsets until there are no itemset can be generated.

3.4 CHAPTER SUMMARY

In general, we see all association rule mining algorithms for semantic data that has been proposed in the literature review define subjects as TID, and (predicate & object) as one pattern. However, in SWARM algorithm the pattern is called common behavior set (CBS).

In semantic association rule mining algorithms, we noted that most of those algorithms did not focus on the confidence between items, they just concentrated on the support for items. Therefore, it is better to use traditional association rule algorithms by considering that the subject as TID while the items as predicate and object together.

All of the above algorithms focused on extracting association rules from semantic data but they did not consider the size of data. However, when the semantic data is too big, those algorithms will not work properly because most of them consist of costly operations such as *intersection* and *join* operations.

“The time complexity of SWARM algorithm belongs to the O(n 2)” (Barati et al., 2017), because SWARM algorithm consists of many loops and join operation to construct semantic association rules, these operations take long time when data is very big.

In next chapter, we will discuss how we can apply semantic association rule on parallel by using Apache Spark framework.

CHAPTER 4

4 Parallel Association Rule Mining Algorithms

4.1 Overview

To date, various methods have been developed and introduced to parallelize association rule mining algorithms in different platforms, whether based on shared memory (Zaïane, El-Hajj, & Lu, 2001), which may cause many synchronizations problems and lower processing speed then poor performance, or based on MapReduce platform (Liang & Wu, 2015; X. Lin, 2014; Makanju et al., 2016; Lijuan Zhou & Wang, 2014; Le Zhou et al., 2010), where data distributed on a number of machines and each part is handled independently.

The association rule mining algorithms are iterative algorithms, because they repeat the scanning of dataset. In Hadoop framework for parallel processing, the data is stored in disk (not in memory) and therefore we need to read data from hard disk storage in each iteration, swapping between memory and permanent storage, and this will take a long time. So, we need another platform that enable us to cache dataset in memory for quick access for repetitive processing.

4.2 Association rule mining on Apache Spark

Several association rule algorithms are implemented in Apache Spark. YAFIM algorithm (Qiu, Gu, Yuan, & Huang, 2014) has been implemented to execute Apriori algorithm in Apache Spark. This algorithm consists of two phases, in the first stage the data is parallelized and then frequent one itemset is generated while (k + 1) itemsets are generated in phase 2. YAFIM algorithm can outperform association rule mining algorithms based on MapReduce such as Apriori based MapReduce in (M.-Y. Lin, Lee, & Hsueh, 2012), because Apache Spark is executed in the main memory and it is more suitable for iterative algorithms. Also, PFP algorithm has been tested by Prasad for large scale data in Apache Spark (Prasad, 2017), the results show that PFP algorithm in Apache Spark outperforms FP-growth algorithm, also PFP algorithm could get high performance on large datasets.

4.3 PFP algorithm in Spark

In Apache Spark, FP-growth algorithm is implemented according to PFP algorithm (H. Li, Wang, Zhang, Zhang, & Chang, 2008) and it can be used in two modules.

4.3.1 PFP based on RDD

First module has been created based RDD object where we have to create RDD from dataset (usually text data) and specify how items are separated (comma, space, or tab characters) and we will also determine the minimum support and the number of partitions. Here, PFP algorithm is implemented by using the methods of RDD object such as map, reduce, count, filter. The result from this module will display frequent itemsets, but not association rules between frequent items.

4.3.2 PFP based on Dataframe

Second module has been created based on dataframe functions such as select, where, join, combineBy, and groupBy.

PFP (parallel FP-growth) algorithm is implemented and used as a ready module, depends on user data type and required results. We will select one of those modules.

Association rules mining is originally used to extract rules from transactions dataset, so data should be stored as a schema as shown in the following table:

Table 4-1 Style of dataset in ARM

Id (any sequential ordered)	**Transactions (categorical data)**
1	I1, I2, I3, I4 (no repeated item in the same transaction)
2	I1,I3

From the above table we may think that data should be stored as tabular format only, but also text data can be converted into dataframe and then apply PFP module based on dataframe directly, or by converting text data into RDD for the module based on RDD.

4.4 Types of data that can be used in association rule mining algorithm

Users can apply association rule mining on different types of dataset formats as follows:

Text dataset, where each line in text is considered as one transaction, and the user have to specify how items are separated from each other, also we need to specify the minimum support and the number of partitions after creating RDD from text dataset. Moreover, Spark enables users to read text datasets as dataframe by importing split from SQL functions module. In split function we will specify how items are separated before we have to create Spark session to create a dataframe, and then apply PFP in the module based on dataframe.

Tabular dataset, where we also have to create Spark session, and then load dataset, which are stored as a data structure in json, csv, or parquet formats, into Spark dataframe and define the schema for dataframe. If we do not define schema, Spark dataframe will infer schema from data structure directly.

4.5 PFP algorithm step by step

The key aspects of PFP algorithm (H. Li et al., 2008) can be listed in following 5 stages:

4.5.1 Sharding

In this stage, the dataset is divided into parts, each part called "shard", and then these shards are distributed on different machines.

4.5.2 MapReduce

In map stage each computer emits (key = item, value = 1), while the mappers finish, the reducers will start to make summation for all values (1s) with the same key. The reducer will emit (key = item, value = support (item)). Besides, the reducer filters out all items that have support less than the minimum support threshold. Distinct items will be stored in F-list in *descending order*, for example if the reducer results are “A”:5, “B”:10, “C”:2, “D”: 6, G:3, H:1 then F-LIST will be as follow:

Table 4-2 F-list

B	D	A	G	C	H

4.5.3 Grouping stage

Group items in F-list by hashing each group into group-id, so that each group will have a unique ID that is called group-id or g-id, and then save the result in G-list. The hash technique is used to facilitate access to each element quickly, subsequently reducing access time to any group.

Actually in machine learning algorithms, we always need to hash data and convert it to numerical data because numbers are lighter than strings. For example, if we want to compare these items “(“value”, “56’)” vs “(“value”, “78”)”, the computer will take long time to compare them; the first comparison will be if v==v, then if a==a, and so on. But, if we assign them to numbers, the comparison will be much faster than separate comparisons.

4.5.4 Parallel FP-growth with MapReduce stages

This stage consists of two parts:

1- Map stage: Each mapper starts with loading G-list in main memory, and then loading the shard of dataset in mapper machine, because one shard consists of the number of transactions, the mapper will start building local Fp-tree as follow:

  I. For each transaction T in the shard, the items will be substituted with corresponding g-id in G-list.

  II. For each g-id in T select K-most right items of this g-id in T and assign those items for this g-id.

2- Reduce stage: The reducer will combine values with the same g-id and calculate support count for values for each g-id.

4.5.5 Aggregation

This step is a final stage, we can explain each stage separately:

I. Map stage: the mapper will just select the g-id with the support greater than or equal to the minimum support and splits elements in values, then feeding the result to reducer.

II. Reduce stage: for each value in values list, the reducer will select the items inside values with the higher support and construct conditional Fp-tree.

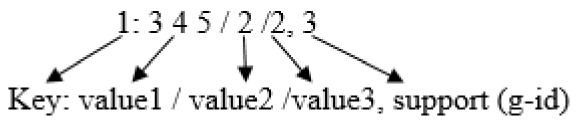
4.6 PFP example

The below figures (**figure4-1**, **figure4-2** and **figure4-3**) show the five steps for PFP algorithm with a simple example by assuming the dataset consists of four transactions: T1, T2, T3, and T4 while each shard consists of two transactions, so each shard is processed by one machine, in parallel counting the mapper will emit (item, 1), while reducer aggregates values with same item and delete items with lower support as D item by assuming minimum support =2, then ordering distinct items in F-list and order items in each transaction according to F-list, then grouping items in F-list and storing them in hash table in this example each character represents one group with unique number called g-id, the next MapReduce to construct local Fp-tree for example in first transaction in figure 4-2 T1=A B C E, we need to do two “for statements” inside map function:

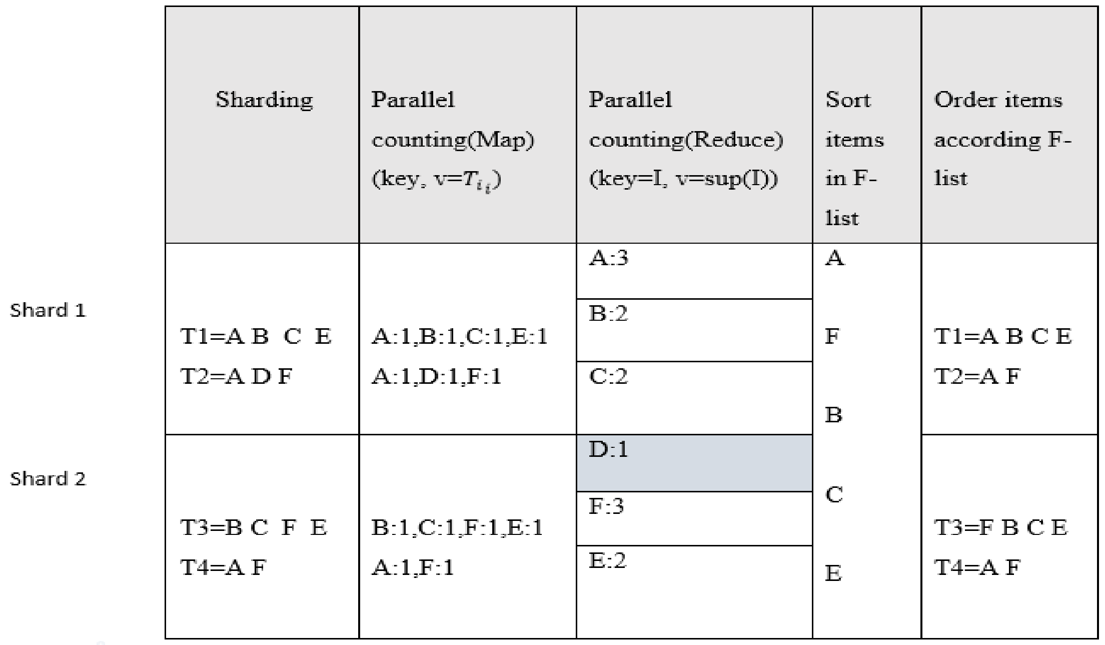
I. First “for statement” to select items inside each transaction one by one, then replace each item with g-id in hash table, T1 will become T1= 1 3 4 5

II. Second for to select each g-id as key and select all right g-ids from this g-id as value, subsequently the result for each mapper will be as following key, value (key: g-id, value: right items from g-id), for T1 the mapper will take 1 as key and all right items 3 4 5 as value, after that 3 as key and 4 5 as value and so on

In reduce stage, each one g-id will go to same reducer, then the reducer will combine all values with same g-id, in this example if g-id =1, all values to this key will be aggregated to this key (1) so 3 4 5 is consider as first value, 2 second value, then 2 is consider third value, therefor the reducer results will be as (key: g-id, sup (v)), when g-id=1, the result become as follow



Finally, the reducer result will be feed to third MapReduce, here the mapper just select g-ids with higher minimum support in our example the successful g-ids are 3, 4, and 1, because the support for all of them equal or higher than 2 threshold, recursively the mapper will emit 1 for each g-id inside each value, the reducer will count all 1s with same g-id and select g-ids that higher than or equal threshold and construct conditional Fp-tree.



	Sharding	Parallel counting(Map) (key, v=T_{i_i})	Parallel counting(Reduce) (key=I, v=sup(I))	Sort items in F-list	Order items according F-list
Shard 1	T1=A B C E T2=A D F	A:1,B:1,C:1,E:1 A:1,D:1,F:1	A:3	A	T1=A B C E T2=A F
			B:2	F	
			C:2	B	
Shard 2	T3=B C F E T4=A F	B:1,C:1,F:1,E:1 A:1,F:1	D:1	C	T3=F B C E T4=A F
			F:3	E	
			E:2		

Figure 4-1 Sharding & MapReduce

Group items (Hashing)	Parallel Fp-growth(map)	Parallel Fp-growth(map)	Parallel Fp-growth(reduce) (key:gid,v,sup(v))
A=1	T1=1 3 4 5	1: 3 4 5 3: 4 5	1: ([3 4 5 / 2 / 2], 3)
F=2	T2=1 2	4: 5 1: 2	3: ([4 5 / 4 5], 2)
B=3			2: ([3 4 5], 1)
C=4	T3=2 3 4 5	2: 3 4 5 3: 4 5	4: ([5 / 5], 2)
E=5	T4=1 2	4:5 1:2	

Figure 4-2 Grouping & second MapReduce

Aggregation (map)	Aggregation (map)	Aggregation (reduce)	Conditional Fp-tree
1: ([3 4 5 / 2 / 2], 3)	1: ([(3:1) (4:1) (5:1) / (2:1) / (2:1)] , 3)	1:[(3:1) (4:1) (5:1) (2:2)]	1: 2
3: ([4 5 / 4 5], 2)	3: [4 5 / 4 5 , 2]	3:[(4:2) (5:2)]	3: 4 5
4: ([5 / 5], 2)	4: [5 5 , 2]	4: [(5:2)]	4: 5

Figure 4-3 Aggregation stage

4.7 Chapter summary

In this chapter, we explained how can we parallelize FP-growth algorithm, however in Apache Spark FP-growth algorithm is paralleled according to PFP algorithm (H. Li et al., 2008). we have explained all stages for PFP algorithm with related example, also we explained two different APIs for PFP algorithm and which type of data can be used on each API.

PFP algorithm in (H. Li et al., 2008) is implemented on MapReduce framework. However, Apache Spark is a best practice for MapReduce because it is executed inside memory; therefore, the implementation of PFP algorithm in Apache Spark is more efficient than Hadoop MapReduce (Afzali, Singh, & Kumar, 2016).

CHAPTER 5

5 Semantic Knowledge Extraction in Big Data

The narrow purpose of this thesis to apply association rule mining on semantic big data. But, the big picture is about knowledge extraction at the age of Internet of Things, big data, and artificial intelligence (AI) in general. In this chapter, we discuss those topics briefly.

5.1 Knowledge hierarchy in the context of IoT

The specific object of this study was to apply ARM in parallel on semantic and big IoT data, and evaluate the performance of those algorithms on such datasets. This means extracting knowledge from IoT data, which has been stored as triples in the format of <subject, predicate, object>. The question is why do we need to mine IoT data, why we are highlighting semantic data instead of raw data for IoT, and what are the benefits of integration of all of semantic web data and the data mining on IoT big data. Before starting, let us explain the knowledge hierarchy for IoT context that has been proposed by Barnaghi, Payam, et al. (Barnaghi et al., 2012) as it is shown below in Figure 5-1 layer by layer starting from the bottom layer.

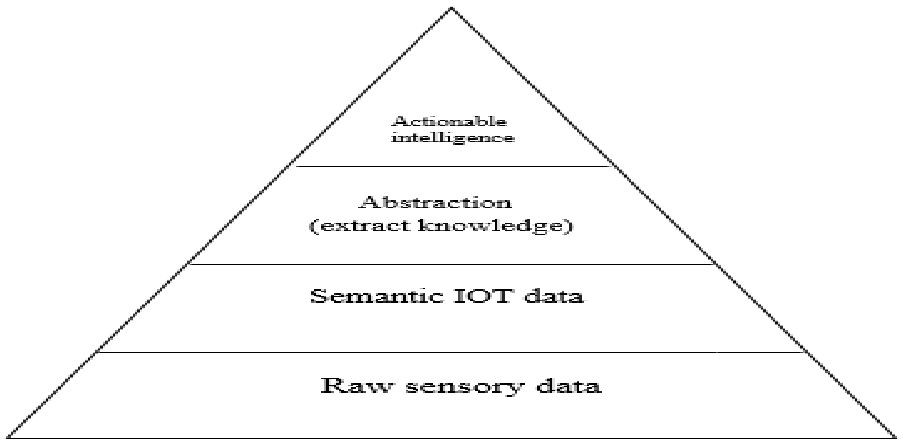


Figure 5-1 Knowledge hierarchy in IoT

5.1.1 Collect raw data

In first layer of the knowledge hierarchy (Figure 5-1) is collecting raw data from IoT devices and sensors. This produces big data from different sensors with different quality and types. For example, temperature data, traffic data, weather data, smart phone data, each of these data domains are different from others. However, IoT data is usually unlabeled data and it does not give us full meaning about data, such as the quality of sensor, the type of resource, the location of resource, or even when the data is taken. So, we need another mechanism to represent data with more semantic information, that is the domain of semantic data. Either sensors generate semantic data, which is not the case today, or the data should be annotated with semantic tags, as it is done in real world applications.

5.1.2 Semantic IoT data

The second layer of the knowledge hierarchy is about semantic IoT data, where data is semantically represented in RDF format or simply as triples (subject, predicate, object). Linked Stream Middleware (LSM) has been used (Soldatos et al., 2015) to project sensor data into RDF graphs.

Although, the semantic Web is not created specifically for IoT applications. But, it has many advantages in the field of IoT applications. Below we describe some of those advantages.

Interoperability

This term is explained by Barnaghi, P, et al. (Barnaghi et al., 2012), which means that semantic web in IoT enables sensors to connect together and share data easily. On the other hand, Korzun, et al (Korzun, Balandin, & Gurtov, 2013) have defined three types of interoperability of smart environments, which include:

- *Interoperability for Device*: to enable devices to discover themselves and get connected properly,
- *Service Interoperability* to share services, for example, send a request to get energy level from a device,
- *Information Interoperability* to make information ready to use from other resources.

Sematic data describes data and metadata

RDF graph gives full description of data such as, location of sensor, time, and quality of data. Therefore, RDF can represent data and metadata in simple and semantic schema. However, the metadata is an important issue in IoT applications, for example, when we get data from temperature sensors, we have to know when the data is taken, which sensor, and location, all this data is essential to make intelligent decision in suitable time. In short, metadata is data describing data, and this is very essential for automatic usage of and knowledge extraction from data flowing from many heterogeneous IoT devices in the age data explosion.

Integration

Semantic web enables the integration among IoT data in various domains. For example, in (Mezghani et al., 2015) different data from different resources could be integrated and share data between them. In the paper (Mezghani et al., 2015) many data resources of wearable sensors are mentioned, such as smart watch, glucose contact lens, heart rate sensors, pressure sensor. All these resources generate different data types and values. By converting this data into RDF graph, resources could easily be integrated together, for example, the integration between heart rate data and pressure data for the same patient could be utilized for further diagnosis.

SPARQL and reasoning

Processing information by using SPARQL queries and reasoning are important advantages in IoT applications. For example, inferring new knowledge by reasoning (Maarala, Su, & Riekki, 2014) allows discovering services to handle resource constrains in IoT devices, or finding compensation mechanisms when resources are unavailable (Barnaghi et al., 2012), or users can query to get for example the speed of wind in specific locations at particular times for specific applications (Calbimonte, Jeung, Corcho, & Aberer, 2012).

Sharing the same data by different applications

Combining the semantic web with the virtualized wireless sensor networks enables using the same sensor data in many other applications (Jafrin, 2015). Although semantic web facilitates connecting different devices, it generates a large graph of data, so we need to mine this graph by using data mining techniques.

5.1.3 Extract knowledge from semantic data

In the third level of the knowledge hierarchy, we extract knowledge from big semantic (RDF) data. With increasing RDF data every day from sensors, this makes RDF data difficult to be handled and understood by machines as a result of much uninteresting or unimportant data generated by devices in large volumes. Integration between data mining and linked semantic data is the optimal solution to this problem (Ristoski & Paulheim, 2016). Most of the recent works focus on applying classification and clustering (Onal et al., 2017) on RDF data, and few of them have used association rule mining, However applying machine learning algorithms on large RDF graph on one machine will cause a bottleneck problem. So, we want a scalable implementation for machine learning algorithms on a suitable platform.

5.1.4 Actionable intelligence

Last layer of knowledge hierarchy is actionable intelligence layer, this layer motivates sensors to make intelligence action, therefore each sensor in IOT should work as smart object that take knowledge from last layer

The specific objective of this thesis is to parallelize FP-growth algorithm on semantic IoT data by the most popular and efficient distributed big data processing framework “Apache Spark”.

5.2 Advantages of using association rule mining on semantic data

There are two primary aims of using association rule mining on semantic data as explained below.

5.2.1 Improving RDF graph

To improve RDF graph as explained in (Abedjan & Naumann, 2011, 2013), researchers have introduced six types of mining RDF graphs for different purposes as shown in Table 5-1.

Table 5-1 Six configurations of context and target

Configuration	context	Target	Purpose
1	subject	Predicate	Schema discovery
2	Subject	Object	Basket analysis
3	Predicate	Subject	Clustering
4	Predicate	Object	Range discovery
5	object	Subject	Topical clustering
6	object	predicate	Schema matching

We can choose one of these configurations depending on the purpose of mining. For example, if we want to discover the schema of data, we will choose one subject as context and we will mine target predicate. Also, if we want to cluster RDF datasets, we can use predicate as context while the target is subject as shown in above Table 5-1.

One purpose in this study is to find the rules between objects when context is predicate. The results show that numbers that frequently occur together in RDF graph model, therefore by using ARM, we can discover the range for sematic data, also this type of mining has pivotal role in improving RDF data and abstracting or summarizing big graph of triples to prevent them from explosion in storage devices, especially in the case of IoT data when it is stored in cloud in order to be analyzed offline.

5.2.2 Mining RDF graph and extracting association rules

To mine RDF graphs and discover relationships between items from many domains, unlike mining market basket analysis, we apply association rule mining on items (sales) and the relation between items is only "buy", and consequently the generated results would describe rules between items that are bought together. In semantic web, we can extract association rules between items, while there are many relationships (predicates) between items. However, semantic data consists of triples (subject, predicate, object), so how can we define items and TID (Transaction Id) in semantic data, if we want to use the whole of data (subject, predicate, object), association rule mining algorithm can mine data with two parameters TID (any id) and items (list of categories).

5.3 Applying association rule mining on semantic data

All of Ramezani, et al. in (Ramezani et al., 2014) and Tsay, et al. (Tsay et al., 2015) have identified subject as TID and (predicate & object) as items, therefore one subject has many predicates and objects for example:

```
:person1 :buy    :milk
:person1 :buy    :banana
:person1 :job    :businessman
:person1 :atTime “15:30”
```

All above triples for only one subject, therefore we can define one transaction as:

```
person1: [(buy,milk),(buy,banana),(job,businessman),(atTime,“15:30”)]
```

For example, if items (buy, milk) & (job, businessman) are frequently occur together, we can say:

“*Most of people that buy milk, they are businessman*”.

Mining semantic web is very important to extract knowledge from large RDF data, because it can be used to extract knowledge from triples with many relationships. However, the item that represents the relation in triple is a predicate, therefore when subject has a unique URI, we can easily make integration between triples in various domains, because URI plays is like a primary key in relational databases (de Medeiros, Priyatna, & Corcho, 2015). As in relational databases we can combine many tables by one primary key, similarly here we can assign one key to many (predicates & objects) tuples in different domains, relate and extract knowledge easily. One of the related works to extract knowledge from many domains is (Mezghani et al., 2015), where machine learning plays an important role to extract facts and detect status of patients either as (anomaly) or (Predicted Anomaly). After detecting the patient state, new knowledge will be added to the facts, and this will help physicians to make decision about diagnosis of patient. This paper (Mezghani et al., 2015) is an application of Knowledge as a Service (KaaS) that runs inside a cloud.

5.4 Apache Spark as cloud to extract knowledge

Apache spark is an open source platform that can run on a cluster of computers. It is suitable for iterative algorithms like machine learning algorithms, because it is running in memory and this avoids the cost of I/O from hard disk storage. FP-growth is one of iterative algorithms that repeats scanning datasets, and then constructing the branches of FP-tree recursively.

In this thesis we converted RDF files to tabular format first, each table having three columns as (subject, predicate, object) respectively. We parallelize triples on the nodes of cluster locally and apply FP-growth in a parallel fashion, consequently extracting frequent patterns from RDF graphs by using Spark. Therefore, Apache Spark can be used to extract knowledge from big data offline in many domains, such as smart cities (Khan, Anjum, Soomro, & Tahir, 2015) (Knowledge as a Service -KaaS).

5.5 Chapter summary

This chapter began by describing the hierarchy of knowledge discovery in context of the IoT applications and explaining the reasons behind representing IoT sensor data as RDF graph model towards greater interoperability and integration among IoT devices and IoT data. Also we discussed how RDF graph model can describe metadata from sensors such as time, location, and quality of sensors in the same graph. We also explained two primary advantages of association rules mining ARM on RDF graph, the first advantage is improving RDF graph, and the second advantage is mining RDF graph and extracting frequent predicates and objects that occur together in large RDF datasets. Also, we explained how can we define transactions on semantic data, and indicated how Apache Spark can be used as cloud to extract knowledge from large semantic data. Therefore, this research is also considered as *Knowledge as a Service* (KaaS) because it extracts knowledge from RDF graph by using Apache Spark platform. Apache Spark platform is an essential component of cloud as service in smart cities (Cheng, Longo, Cirillo, Bauer, & Kovacs, 2015).

The next chapter describes the procedures and methods used in this work.

CHAPTER 6

6 Implementation

In this chapter, we present the full description for our methodology. We begin with how we can parse RDF graphs to tabular format first, and then we explain how we handle and parse blank nodes (*Bnode*) in semantic data. After parsing RDF graphs to a table of three columns, we select important features, and then process data and handle missing values. We then convert data to a specific schema, and finally we apply FP-growth algorithm on this data. We also implement SAG algorithm (Tsay et al., 2015) by using SQL queries in PySpark data frame.

6.1 Implementation of FP-growth on semantic data using Apache Spark

In this section, we explain how we extract frequent patterns from triples. First we explain how triples are converted to tabular format, and second, we describe the hierarchy of applying machine learning step by step as follow:

I. Feature selection.
II. Preprocessing data.
III. Transformation.
IV. Last chapter convers evaluation and interpretation

6.2 Serialization of RDF graph into CSV file

RDF (Resource Description framework) is a data model to represent semantic data graphs. This graph data can be materialized in different data formats such as RDF/XML, TURTLE, or N3 among some other data formats. However, Apache Spark does not support RDF data type formats directly. Therefore, we need to serialize this graph to a table format so that it can be processed in Apache Spark platform. In this study we used RDFLIB library that enables us to handle RDF graph formats in Python programming language, and also it enables us to query data, call triples, and then store them in Comma Separated Values (CSV) file format.

6.3 Types of RDF data formats used

RDF/XML Data Format

First type used is RDF/XML. This data format is used in datasets in NYC OpenData website[5], one of the data sources we have used. The site includes open data in various data formats in different domains such as business, education, environment, and health, and more. Those datasets are provided by many agencies of New York City, for example, HHC (Health and Hospital Corporation), NYCHA (New York City Housing Authority) and many other agencies. Since most of the datasets are easy to read and understand, we just simply parse them into CSV files.

NYC OpenData website contains data that is collected from New York City. New York City is considered one of the smartest cities in the world. We can apply ARM on datasets from this site and extract frequent patterns easily. **figure 6-1** below shows a sample of RDF/XML data, which is collected by Department of Health and Mental Hygiene (DOHMH) Agency in New York City.

[5] https://opendata.cityofnewyork.us/data/

```
<rdf:Description
rdf:about="https://data.cityofnewyork.us/resource/jb7j-
dtam/2189">
    <socrata:rowID>2189</socrata:rowID>
    <rdfs:member
rdf:resource="https://data.cityofnewyork.us/resource/jb7j-
dtam"/>
    <ds:year>2010</ds: year>
    <ds:leading_cause>Assault(Homicide:Y87.1,X85-Y09)
    </ds:leading_cause>
    <ds:sex>M</ds:sex>
    <ds:race_ethnicity>Black                           Non-
Hispanic</ds:race_ethnicity>
    <ds:deaths>299</ds:deaths>
    <ds:death_rate>35.1</ds:death_rate>

<ds:age_adjusted_death_rate>35.5</ds:age_adjusted_death_rat
e></rdf:Description>
```

Figure 6-1 RDF/XML NYC dataset example

Above RDF/XML is then converted into a table of three columns as triples (Table 6-1), and saved as a CSV file.

Table 6-1 Store triples in csv file

Subject	predicate	object
https://data.cityofnewyork.us/resource/jb7j-dtam/2189	:rowID	2189
https://data.cityofnewyork.us/resource/jb7j-dtam/2189	:member	https://data.cityofnewyork.us/resource/jb7j-tam
https://data.cityofnewyork.us/resource/jb7j-dtam/2189	:year	2010
https://data.cityofnewyork.us/resource/jb7j-dtam/2189	:leading_cause	Homicide: Y87.1, X85-Y09
https://data.cityofnewyork.us/resource/jb7j-dtam/2189	:sex	M
https://data.cityofnewyork.us/resource/jb7j-dtam/2189	:deaths	299
https://data.cityofnewyork.us/resource/jb7j-dtam/2189	:death_rate	35.1

Please note that the subjects and predicates are always URI values, while objects can be either URI or literal values. In order to shorten the values, we took only the last part of URI, therefore for example the subject URI can be :2189 instead of **https://data.cityofnewyork.us/resource/jb7j-dtam/2189**, and this conversion can be done easily in a scripting language or as Excel functions in Microsoft Excel as follows:

```
=RIGHT(domain,LEN(domain)-
FIND("*",SUBSTITUTE(domain,"/","*",LEN(domain)-
LEN(SUBSTITUTE(domain,"/","")))))
```

TURTLE Data Format

Second data type is TURTLE. CITYPULSE website[6] is another source of data we have used, and it has datasets in Turtle format. This data website contains many datasets for smart city domain, such as traffic dataset, pollution dataset, weather dataset. All datasets are available in two formats, the first is raw data that is stored in CSV files, and the second is semantic data that is stored in Turtle format.

In our study, we used traffic dataset and pollution dataset, text bellow shows a sample of this data for traffic dataset in Turtle format.

[6] http://iot.ee.surrey.ac.uk:8080/datasets.html

```
<http://iot.ee.surrey.ac.uk/citypulse/datasets/traffic/traffic_oct_nov/trafficData158355#observations_av
gTime_qkqf4fai8kp5ntgk0ohgljlpp> a sao:Point ;

        sao:hasUnitOfMeasurement unit1:seconds ;

        ns1:featureOfInterest
<http://iot.ee.surrey.ac.uk/citypulse/datasets/traffic/traffic_oct_nov/trafficData158355#context_ksh9bc
1gc46nrq1rtm7uf6t62r> ;

        sao:time [a tl:Instant;

                tl:at "2014-10-04T07:40:00"^^xsd:dateTime ];

        sao:value "52" .

<http://iot.ee.surrey.ac.uk/citypulse/datasets/traffic/traffic_oct_nov/trafficData158355#context_ksh9bc
1gc46nrq1rtm7uf6t62r>  a sao:FeatureOfInterest ;

        ct:hasFirstNode [ a ct:Node ;

                ct:hasNodeName "REPORT ID: 158355" ] .
```

Figure 6-2 TURTLE data sample from CityPulse Dataset

Although the above Turtle graph seems clear, when we store it in a CSV file in the form of (subject, predicate, object), we face some complexities due to data containing blank nodes (Bnode). Actually, there are two reasons to use Bnodes in a graph data model (Segaran, Evans, & Taylor, 2009):

1. If we do not have a URI to refer to a thing because there is no available identifier for this thing. But, that does not mean we cannot talk about this thing, we can use Bnode instead of a URI.
2. If we want to group a set of statements in the same node, we can use Bnode to combine statements, such as the following Bnode example:

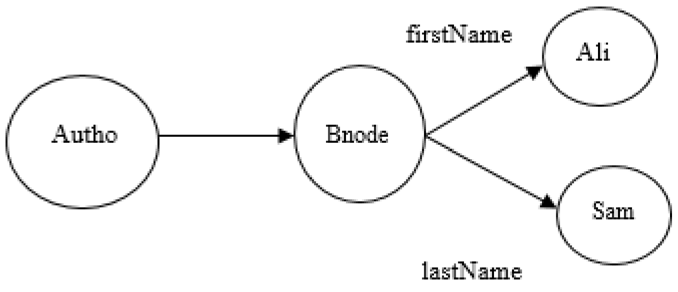


Figure 6-3 Bnode Example

In above Turtle serialization (Figure6-3), the first Bnode is used to combine time and type in the same node. We can represent Turtle dataset in **figure 6-2** as a graph that is shown in figure 6-4. The graph has nine classes, two of them are Bnodes, no Bnodes have a URI because they use Bnode to combine two statements such as:

(:sensor :a :instant) & (:sensor :at “2014-10-04T07:40:00”)

We notice that all of subject, predicate, and object values have a URI in the first statement, while the object in the second statement is a literal, so it is not URI.

The figure below represents only one observation for one sensor at specific time, however if you have many sensors, the graph will be a very big graph, and this requires a large storage and a scalable processing solution is needed.

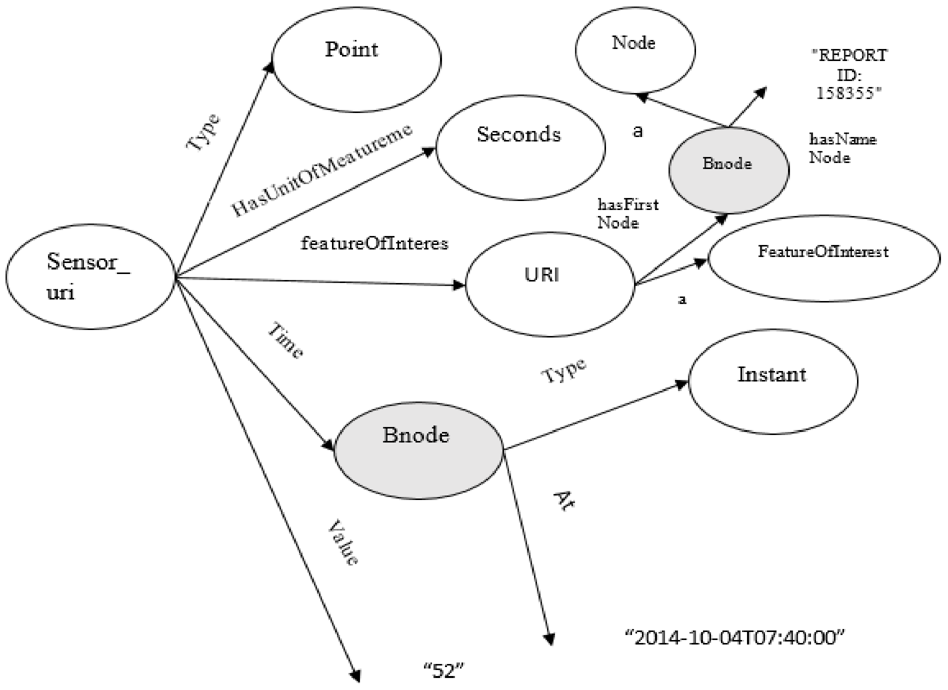


Figure 6-4 RDF graph data example from CityPulse dataset

6.4 Combine values that occur at same time

In the above graph (**Figure 6-4**), we are interested in mining values that occur at the same time. When predicate is time, the object is Bnode, this Bnode is subject for all of “at” and “type” together, therefore all of “at” and “type” are sharing the same subject but we do not need type here, we only need “at” in order to know when this value is taken by sensor, we follow those steps to combine values that occur at same time. All values that are occurred at same time are stored as one transaction.

First, we store RDF graph as a CSV file of three columns, the object for “time” predicate is subject for “at” predicate, and also the same sensor (subject) has value and time. We want to store all of time and value at the same row. The following table shows the result after we store the graph into a table, and filter “at”, “time” and “value” predicates.

Subject	predicate	Object
Sensor_uri	value	52
Sensor_uri	time	Bnode
Bnode	at	2014-10-04T07:40:00

Second, we join triples where object (Bnode) = subject (Bnode)

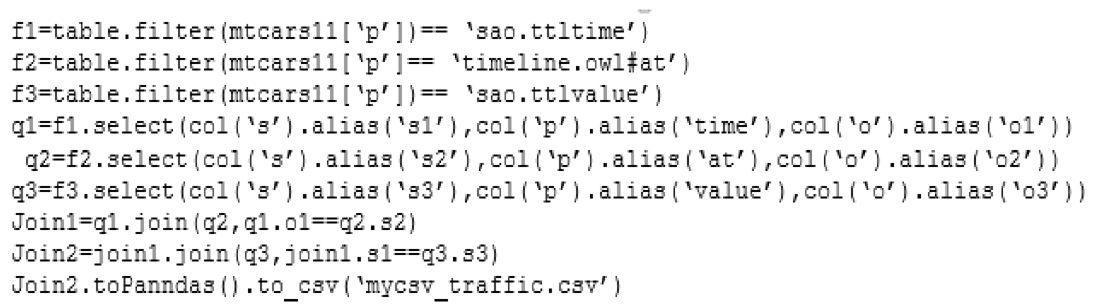
s	p	o	S	P	O
Sensor_uri	time	Bnode	Bnode	at	2014-10-04T07:40:00

Third, we separate time from date and we order data by date then by time

s	p	O	p	Date	Time
Sesor_uri	value	52	at	2014-10-04	07:40:00

Finally, we combine between all values that occur at same time as one transaction.

The following code show “join” operation that is used in this example. We use SQL queries in PySpark data frame, and in some cases we join values that occur at the same hour. Then, we combine values that occur at the same week, such as culture events dataset when we mined the frequent events that occurred in the same week.



```
f1=table.filter(mtcars11['p'])== 'sao.ttltime')
f2=table.filter(mtcars11['p']== 'timeline.owl#at')
f3=table.filter(mtcars11['p'])== 'sao.ttlvalue')
q1=f1.select(col('s').alias('s1'),col('p').alias('time'),col('o').alias('o1'))
 q2=f2.select(col('s').alias('s2'),col('p').alias('at'),col('o').alias('o2'))
q3=f3.select(col('s').alias('s3'),col('p').alias('value'),col('o').alias('o3'))
Join1=q1.join(q2,q1.o1==q2.s2)
Join2=join1.join(q3,join1.s1==q3.s3)
Join2.toPanndas().to_csv('mycsv_traffic.csv')
```

Code 6-1 Restore Bnode into CSV file

In **Code 6-1**, we first filter triples in which predicates are ‘time’, the second line to filter triples that predicates are equal to ‘at’, third line to filter triples that their predicates are equal to ‘value’. All of next three lines to rename subject, predicate, and object. The last three lines to join triples in which objects are equal to subjects, then storing the result in CSV file.

6.5 Association rules mining steps

We followed steps of data mining to mine RDF graph. First step is feature selection, second step is preprocessing data, third step is transformation step. Finally, apply PFP algorithm on dataset (**Figure 6-5**).

6.5.1 Feature selection

We selected only data that can be frequent and ignored others, for example in above RDF/XML file, we ignored rowID, because it must not be repeated. Actually select important features is a very important step in data mining; because this step reduces the size of data and reduce time implementation.

6.5.2 Preprocessing data

In preprocessing step, we converted continuous numeric data to categories, Apache Spark facilitates grouping items by using Bucketizer module that is available in pyspark.ml.features package, which contains many modules to handle data. This step is very important when numeric data is continuous, because the nature of association rules algorithm is about extracting rules from categorical data but not continuous numeric data. If we do not convert numbers to categories, the number of generated rules will be very small when minimum support is large, on the other hand, if minimum support is small, the number of rules will be very large.

6.5.3 Transformation

In this step, we create RDD from triples that are stored as a table of three columns. Then, by using combineByKey (one of RDD object methods in PySpark), we combine all (predicates & objects) that have the same subject; the subject here is used as a key to combine items (predicate &object). This step is very important because the association rule mining algorithms can handle data that have specific schema (TID and items). Here, TID is the subject, while items are the pairs of (predicate, object).

6.5.4 Evaluation and interpretation

In this step, we feed data to FP-growth algorithm (dataframe based API), and we get results as a dataframe. After that we made some queries on ARM model and extract preferred rules.

Figure 6-1 summarizes our methodology step by step.

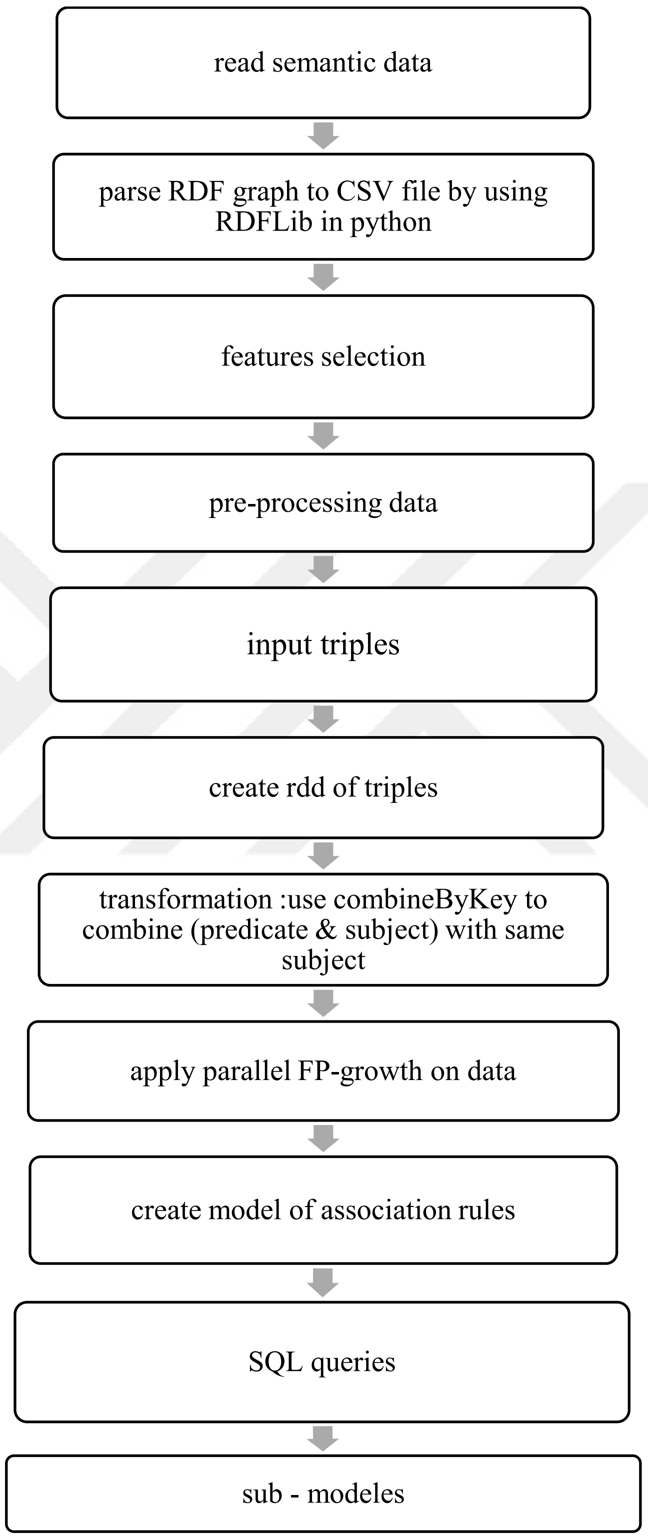

read semantic data
parse RDF graph to CSV file by using RDFLib in python
features selection
pre-processing data
input triples
create rdd of triples
transformation :use combineByKey to combine (predicate & subject) with same subject
apply parallel FP-growth on data
create model of association rules
SQL queries
sub - modeles

Figure 6-5 Proposed methodology

6.6 SAG algorithm by using SQL queries

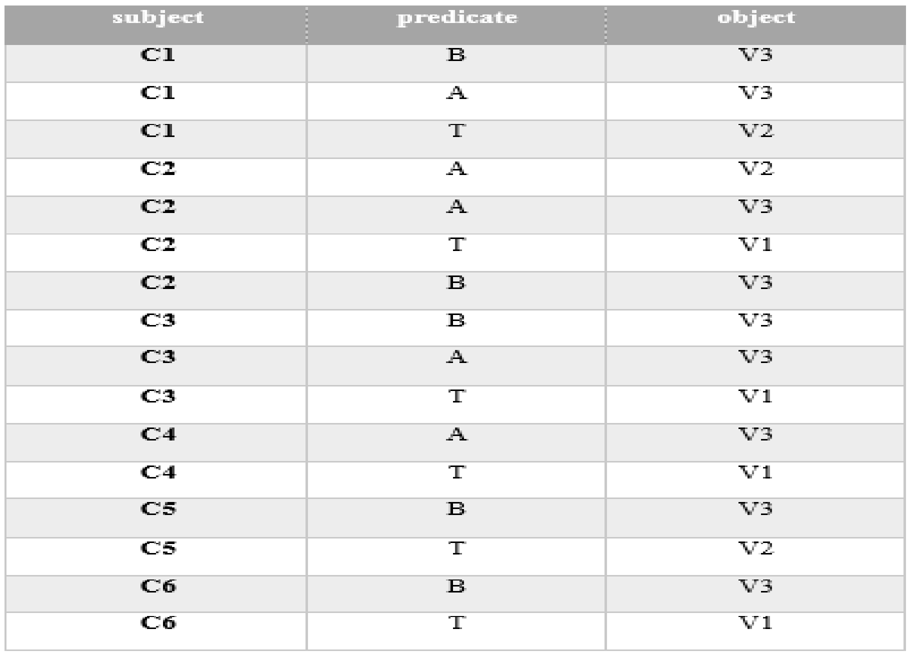
SAG algorithm is proposed by Tsay et.al (Tsay et al., 2015). This algorithm can extract association rules from RDF graphs by using SPARQL queries. Therefore, SAG algorithm can work directly on semantic datasets without converting it to a tabular format. The idea of SAG algorithm is explained in Chapter 3.

One of our attempts in this thesis is applying SAG algorithm by using SQL queries in PySpark dataframe. We tried matching between SPARQL queries that have been used in SAG algorithm and the corresponding SQL queries, depending on some literature reviews that have used SQL queries in Spark to execute SPARQL queries (Naacke, Curé, & Amann, 2016; Schätzle, Przyjaciel-Zablocki, Skilevic, & Lausen, 2016). The following example explains SAG algorithm by using SPARQL queries and how we have used SQL for each SPARQL query.

6.6.1 SAG algorithm Example

Assume you have the following RDF dataset (**Table 6-2**), and the minimum support is 2.

Table 6-2 Triples (s, p, o)



subject	predicate	object
C1	B	V3
C1	A	V3
C1	T	V2
C2	A	V2
C2	A	V3
C2	T	V1
C2	B	V3
C3	B	V3
C3	A	V3
C3	T	V1
C4	A	V3
C4	T	V1
C5	B	V3
C5	T	V2
C6	B	V3
C6	T	V1

First, determine target predicate that you want to extract association rules for it, in this example the target predicate is ‘T’. actually you can choose any target predicate. In patient dataset, you may interest to extract ARM when predicate equal “headache”, the other person want to mine “cancel” predicate; therefore, you are free to determine any interested predicate.

Second, calculate support count for ‘T’ and object together as one item, while the target predicate may have many objects. So, we will filter all items that have a support greater than or equal to the minimum support.SPARQL1:

```
Select ?p ?o (COUNT(*) AS ?sup)
WHERE {?s ?p ?o.
     FILTER(regex(str(?p), ‘T’ , ‘I’)).}
GROUP BY ?p ?o
HAVING(?sup>= min_sup).
```

Code 6-2 Calculate the support count for target predicate

The corresponding Python code is shown below:

```
q=table.select('s','p','o').where(mtcars1['p']=='T'
)
q1=q.groupBy('p','o').count()
q1.filter(q1['count']>= min_sup).show()
```

We get the same result from the above two queries:

Table 6-3 Result1 of SAG algorithm (one-factor set)

p	o	Count
T	V2	2
T	V1	4

Third, scan the dataset and filter all other non-target predicates that have support greater than or equal to the minimum support. SPARQL query for this is as follows:

SPARQL2:

```
Select ?p ?o (COUNT(*) AS ?sup)
WHERE {?s ?p ?o.
     FILTER(!regex(str(?p), ‘T’ , ‘I’)).}
GROUP BY ?p ?o
HAVING(?sup>= min_sup).
```

Code 6-4 Calculate the support count for non-target predicate

And Python code the query is as follows:

```
q2=table.select('s','p','o').where(mtcars1['p']!='T')
q3=q2.groupBy('p','o').count()
q3.filter(q1['count']>= min_sup).show()
```

Code 6-5 Calculate the support count for non-target predicate

The result of the above two queries are:

Table 6-4 Result of SAG algorithm (one factor set (support count for predicates))

P	o	Count
A	V3	4
B	V3	5

Fourth, we combine SPARQL1 & SPARQL2 to construct 1-factor-sets as with the following SPARQL query:

```
SELECT ?p1 ?o1 ?p2 ?o2 (COUNT(*) AS ?sup)
WHERE {?s ?p1 ?o1.
        FILTER (regex(str(?p1), ‘T’, ‘i’)).}
                ?s ?p2 ?o2
        FILTER (!regex(str(?p2), ‘T’, ‘i’)).}
GROUP BY ?p1 ?o1 ?p2 ?o2
HAVING (?sup>=2)

ORDER BY ?p2 ?o2
```

Code 6-6 Two-factor set

The above query consists of two queries, the set of subjects from first query will intersect with the second query. However in SQL language this query consists of many queries, also we need to define UDF (user define function) for intersection operation.

SQL3-1

```
w=table.select('s','p'
,'o').where(mtcars1['p']=='T')
w1=w.groupBy('p','o').agg(collect_list('s')).show()
```

Code 6-7 First step to generate two-factor set

Result

Table 6-5 the result of **code 6-7**

p	o	Collect list
T	V2	[C1,C5]
T	V1	[C2, C3, C4, C6]

SQL3-2

```
w2=table.select('s','p'
,'o').where(mtcars1['p']!='T')
w3=w2.groupBy('p','o').agg(collect_li
st('s')).show()
```

Code 6-8 Second step to generate two-factor set

Result

Table 6-6 the result of **code 6-8**

P	o	Collect list
A	V3	[C1, C2, C3, C4]
B	V3	[C1, C2, C3, C5, C6]
A	V2	[C2]

UDF(conditional intersection function)

```
import pyspark.sql.functions as f
from pyspark.sql.types import *
intersection_udf  =  f.udf(lambda  u,  v:  list(set(u)  &
set(v)), ArrayType(StringType()))
intersection_length_udf = f.udf(lambda u, v: len(set(u) &
set(v)), IntegerType())
```

Code 6-9 User defined function (conditional intersection)

Cross join operation to generate 2-factor-set

```
df1.alias("l")\
    .crossJoin(df2.alias("r"))\
    .select(
        f.col('l.p').alias('lp'),
        f.col('l.o').alias('lo'),
        f.col('r.p').alias('rp'),
        f.col('r.o').alias('ro'),
        intersection_udf(f.col('l.c'),
f.col('r.c')).alias('TID'),
        intersection_length_udf(f.col('l.c'),
f.col('r.c')).alias('len')
    )\
    .where(f.col('len') > 1)\
    .select(
        f.struct(f.struct('lp', 'lo'),
f.struct('rp', 'ro')).alias('2-Itemset'),
        'TID'
    )\
    .show()
```

Code 6-10 Cross join

Result

Table 6-7 two-factor set

2-itemset	TID
[[T, V2], [B, V3]]	[C1, C5]
[[T, V1], [A, V3]]	[C3, C2, C4]
[[T,V1], [B, V3]]	[C3, C2, C6]

Unfortunately, this code took very long time to execute, so we decided to stop writing SQL code, SAG algorithm is difficult to run by using SQL queries; because this algorithm requires *join* and *intersection* operations. Both of these operations are costly and take very long time , on this field Naake, et.al (Naacke et al., 2016) has proposed appropriate methods to optimize *join* algorithms that used in SPARQL language in Apache Spark platform.

6.6.2 Advantages of SAG algorithm

- Extract association rules from RDF graph directly.
- It works well when dataset is small.
- Enable mining graph by using SPARQL queries.

6.6.3 Disadvantages of SAG algorithm

- It is costly, because it repeats scanning the whole dataset in each query (costly I/O)
- It does not work well when dataset is big
- It loses its scalability, when it run in parallel
- It uses join operation which is consider the most expensive operation (Naacke et al., 2016)

Due to limitations of SAG algorithm that uses SPARQL queries in mining graph, we can conclude that mining RDF graph by using SPARQL language can be used in limited size of data.

6.7 CHAPTER SUMMARY

In this chapter, we covered most of implementation steps. First we parse RDF graph to CSV file where the data is stored as three columns (subject, predicate, and object), after that we select important features and ignore non-frequent items such as ID. In preprocessing step, we convert continuous numeric data to categories, for example number 0 of age refers to first category of age [10, 15), also number 1 refers to second category [15, 25) and so on. In preprocessing step we also replace the missing values by the mean, for example in Pima dataset, there are many missing values, we handle this problem by calculate the mean of each column, then we replace missing values by the mean.

In transformation step, we combine all predicates and objects that have same subject, this step enable us to apply PFP algorithm directly on our data.

Table 6-8 The dataset after transformation step

TID	Items
subject	$[(p_1, o_1), (p_1, o_2), (p_2,\ o_3), (p_3, o_4)]$

We apply PFP on our dataset, the ARM that generated describe frequent predicate and object that occur frequently in dataset, we make SQL queries on our model to extract preferred predicate and object.

CHAPTER 7

7 Evaluation

In this chapter, we will evaluate PFP (Parallel Fp-growth) and SAG (Semantic Association Generator) algorithms in Apache Spark framework. First, we will explain the datasets that are used, also we give full description for dataset such as number of triples, type of data and download site. We display ARM for each data , then we apply some queries on ARM model. Finally, we refers to advantages of ARM on each dataset.

7.1 Overview of test datasets

The datasets to be tested in machine learning algorithms are very important. We used a number of different semantic datasets from two sources, namely NYC OpenData [5] and CityPulse[6]. These resources contain large numbers of annotated datasets in various domains for especially smart city applications. We believe that using real datasets is better than simulating; because real data can make algorithms learn from the actual human activities or environments. However, the same algorithms can learn different behaviors from different environments. We will explain each dataset and advantages of using ARM on it. The following table (**Table** 7-1) gives a summary of the datasets we used in this study.

Table 7-1 datasets

Dataset	Source	Number of triples	Agency publishing
Public Pay Phone Locations	NYC OpenData	17,384	Telecommunication and Information Technology
Traffic Volume Count	NYC OpenData	30460	Department Of Transportation in NYC
Metal Content of Consumer Products	NYC OpenData	6500	Department of Health and Mental Hygiene (DOHMH)
Culture Event Dataset	CityPulse	1391	-
Road Traffic Dataset	CityPulse	340961	-
Pollution	CityPulse	340961	-
Smart Meters in London	Kaggle	232816	-
Pima Dataset	Kaggle	6913	-

7.2 NYC OpenData dataset

This first dataset is from NYC open data website[5]. This website contains different datasets in many domains such as city government, environment, business, education, and health domains. These datasets are collected by many agencies in New York City. It is open data and available in CSV, RDF/XML, and other open data formats. Samir Saini is the Commissioner, who is working at NYC Information Technology & Telecommunications. He said “The NYC Open Data portal is a powerful tool that ensures transparency and fosters civic innovation within our City to help improve the quality of life for all New Yorkers.”, this means that data can contribute to the development of New York City. Supervisor machine learning algorithms always need training data to learn and develop ML models. ARM algorithm is also good candidate to test on this data for specific problems.

From NYC Open Data we use three datasets and apply ARM on them, the following part will explain each dataset and present the results of the tests.

7.2.1 Public Pay Telephone Locations dataset

Data format: RDF/XML

Download site: https://data.cityofnewyork.us/Social-Services/Public-Pay-Telephone-Locations/5qrc-eb74

Dataset name: Public Pay Telephone Locations

Agency publishing: Telecommunication and Information Technology

Number of triples: Graph has 17384 statements.

Sample of dataset

```
<rdf:Description
rdf:about="https://data.cityofnewyork.us/resource/_5qrc-
eb74/1">
    <socrata:rowID>1</socrata:rowID>
    <rdfs:member
rdf:resource="https://data.cityofnewyork.us/resource/_5qrc
-eb74"/>
    <ds:company_name>CityBridge LLC</ds:company_name>
    <ds:installation_id>8821</ds:installation_id>
    <ds:ppt_id>125749</ds:ppt_id>
    <ds:active_indic>Y</ds:active_indic>
        <ds:number_of_phones>1</ds:number_of_phones>
```

Figure 7-1 RDF/XML PPTs dataset

As we see from above RDF graph, the dataset contains infrequent items such as ppt_id, we filtered out all infrequent items before applying ARM on it.

Number of frequent columns after filtering: 7554.

Minimum support=0.1

Minimum confidence=0.1

Association rules from public pay telephone locations dataset

Table 7-2 ARM for PPTs

	antecedent	consequent	confidence
1	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('cluster_position', 'S1'), ('mounting', 'PEDESTAL')]	[('wi_fi', 'N ')]	1
2	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('cluster_position', 'S1'), ('mounting', 'PEDESTAL')]	[('active_indic', 'Y')]	1
3	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('cluster_position', 'S1'), ('mounting', 'PEDESTAL')]	[('company_name', 'CityBridge LLC')]	1
4	[('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('advertising', 'N')]	0.43
5	[('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('wi_fi', 'N ')]	1
6	[('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('tax_lot', '1')]	0.2
7	[('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('councilmanic_district', '38')]	0.17

Number of rules 5046

The above extracted rules can be explained below:

First rule R1 means “most of PPTs with no advertising space and, enclosure type “SMALL” and mounting type 'PEDESTAL, do not have Wi-Fi service”.

After we created the model of association rules between items, we can store this model as data frame, also we can make some queries on this model in order to find specific rules or avoid some rules, actually we did four queries on this model by determining items that we want to extract rules for them.

User defined items

Semantic association rule mining results consist of many rules, some of these rules do not give us full meaning about the dataset, consequently we enable users to determine their preference items or avoid others by using SQL queries.

7.2.2 Queries

Query1: display all association rules for PPTs that do not have wi-fi service and they are active status then order the rules by confidence in descending order.

```
que1=r11[r11.antecedent.apply(lambda  x  :  [(Wi-Fi,  N),
(active_indic, Y)].issubset(x) )]
```

Code 7-1 Extract I1 AND I2 from ARM model

Result:

Table 7-3 Sub-model1 for PTTs

	antecedent	consequent	confidence
2523	[('enclosure_type', '4-Mar'), ('cluster_position', 'S1'), ('wi_fi', 'N '), ('active_indic', 'Y')]	[('company_name', 'CityBridge LLC')]	1.00
2945	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('cluster_position', 'S1'), ('wi_fi', 'N '), ('active_indic', 'Y')]	[('company_name', 'CityBridge LLC')]	1.00
2956	[('tax_lot', '1'), ('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N '), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('cluster_position', 'S1')]	1.00
2971	[('enclosure_type', '4-Mar'), ('advertising', 'Y'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N '), ('active_indic', 'Y')]	[('company_name', 'CityBridge LLC')]	1.00
3065	[('tax_lot', '1'), ('cluster_position', 'S1'), ('number_of_phones', '1'), ('wi_fi', 'N '), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('mounting', 'PEDESTAL')]	1.00
3010	[('enclosure_type', '4-Mar'), ('cluster_position', 'S1'), ('number_of_phones', '1'), ('wi_fi', 'N '), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('mounting', 'PEDESTAL')]	1.00

We ordered result by confidence in descending order, the whole number of rules is about 516

Query2 display all rules for PPTs that do not have advertising space nor wi-fi.

If [(Wi-Fi, N) AND (advertising, N) then?]

Result in descending order

Table 7-4 Sub-model2 for PPTs ARM

	antecedent	consequent	confidence
2957	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('wi_fi', 'N ')]	[('active_indic', 'Y')]	1.00
2945	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('cluster_position', 'S1'), ('wi_fi', 'N '), ('active_indic', 'Y')]	[('company_name', 'CityBridge LLC')]	1.00
2958	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('wi_fi', 'N ')]	[('company_name', 'CityBridge LLC')]	1.00
2972	[('advertising', 'N'), ('cluster_position', 'S1'), ('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N ')]	[('active_indic', 'Y')]	1.00
2973	[('advertising', 'N'), ('cluster_position', 'S1'), ('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N ')]	[('company_name', 'CityBridge LLC')]	1.00
2916	[('enclosure_type', 'SMALL'), ('advertising', 'N'), ('cluster_position', 'S1'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N ')]	[('active_indic', 'Y')]	1.00

Number of rules: 201

Query 3: display all rules when antecedent is enclosure type “SMALL” **union** “enclosure type 4-ar”

IF(enclosure_type, SMALL) OR (enclousure_type, 4-mar) THEN ?

Number of rules is 1008

Result in descending order

Table 7-5 Sub-model for PTTs ARM

	antecedent	consequent	confidence
504	[('enclosure_type', '4-Mar'), ('advertising', 'Y'), ('cluster_position', 'S1'), ('wi_fi', 'N ')]	[('active_indic', 'Y')]	1.00
353	[('enclosure_type', '4-Mar'), ('cluster_position', 'S1'), ('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('company_name', 'CityBridge LLC')]	[('active_indic', 'Y')]	1.00
561	[('enclosure_type', '4-Mar'), ('advertising', 'Y'), ('cluster_position', 'S1'), ('number_of_phones', '1'), ('active_indic', 'Y')]	[('mounting', 'PEDESTAL')]	1.00
562	[('enclosure_type', '4-Mar'), ('advertising', 'Y'), ('cluster_position', 'S1'), ('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N '), ('company_name', 'CityBridge LLC')]	[('active_indic', 'Y')]	1.00
563	[('enclosure_type', '4-Mar'), ('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N '), ('active_indic', 'Y')]	[('cluster_position', 'S1')]	1.00
564	[('enclosure_type', '4-Mar'), ('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('wi_fi', 'N '), ('active_indic', 'Y')]	[('company_name', 'CityBridge LLC')]	1.00

We create function to run above query. This function called OR function.

UDF (User Defined Function (OR))

```
from pyspark.sql import functions as F
from pyspark.sql import types as T
def checkIsIn(array):
    return True in [x in [4,3] for x in array]
udfCheckIsIn = F.udf(checkIsIn, T.BooleanType())
```

Code 7-2 Extract I1 OR I2 from ARM model

Query (code)

```
r1=model.associationRules

qf1 = r1.filter(udfCheckIsIn(r1.antecedent))

qf1.toPandas()
```

Code 7-3 Filter I1 OR I2 from antecedent

In the last line of above code, we convert data frame to pandas data frame in order to display result clearly.

Query4: display all rules for PPTs that have advertising space.

IF (antecedent=?) then (‘advertising, Y’)

Result

Table 7-6 Sub-model for PPTs ARM

	antecedent	consequent	confidence
8	[('number_of_phones', '1'), ('mounting', 'PEDESTAL'), ('active_indic', 'Y'), ('company_name', 'CityBridge LLC')]	[('advertising', 'Y')]	0.57
17	[('enclosure_model', 'ADVLinkNYC'), ('cluster_position', 'LK')]	[('advertising', 'Y')]	1
39	[('enclosure_type', 'Link'), ('mounting', 'Kiosk'), ('wi_fi', 'Y '), ('number_of_phones', '1')]	[('advertising', 'Y')]	1
44	[('number_of_phones', '2')]	[('advertising', 'Y')]	0.65
71	[('enclosure_type', '4-Mar'), ('cluster_position', 'S1'), ('mounting', 'PEDESTAL'), ('active_indic', 'Y')]	[('advertising', 'Y')]	0.98
87	[('enclosure_type', '4-Mar'), ('cluster_position', 'S1'), ('active_indic', 'Y')]	[('advertising', 'Y')]	0.98

Number of rules = 416

7.2.3 Benefits of using ARM on Public Pay Telephone Location dataset

7.2.3.1 ARM as classifier

ARM plays an essential role to give us overview about PPTs and relationships between items, in this dataset, we notice that the results show correlation between different types of item, as a result we get full understanding of PPTs inside New York City, consequently these facts and knowledge will contribute in city planning and management, besides these rules can be used to make classifications between PPTs in New York, for example, if you want to classify PPTs that have advertising space from others that do not have advertising space, therefore when we are setting consequent to be either (Advertising, Y) or (Advertising, N), we can classify PPTs and ARM can work as classifier, this method has been used by (Ma & Liu, 1998) the result showed that using association rule algorithm can improve quality of classifier, more recently work and in field IOT is analysis of human activities by using both of association rule and classification (Atzmueller, Hayat, Trojahn, & Kroll, 2018).

7.2.3.2 ARM in feature selection stage

Association rule mining can select important items from large dataset for example, in above association rules result, we notice that many items are repeated together with confidence 100%, in this state, we have to select one of them and ignore others, therefore ARM can be used at feature extraction step and before classification step.

7.2.4 Traffic Volume Count Dataset

This dataset is collected by DOT (Department Of Transportation) for New York

Data format: RDF/XML

How it is collected: by sensors that are put on along streets

Download site: https://data.cityofnewyork.us/Transportation/Traffic-Volume-Counts-2011-2012-/wng2-85mv/data

Dataset name: Traffic volume counts

Number of triples: graph has 30460 statements.

Number of frequent columns: 27458.

Description of data: this data shows number of traffic in each hour for one street segment, however this data is represented as semantic data or as triples (subject, predicate, and object) each subject represent resource URI for each sensor, while predicate is either hours, dates, street name or direction as shown below.

We try to find frequent number of vehicles at each hour in all directions. For example in below RDF graph, the trip begins from UNION PLACE to VAN DUZER STREET, this trip at NB direction.

```
<rdf:Description
rdf:about="https://data.cityofnewyork.us/resource/wng2-
85mv/1">
    <socrata:rowID>1</socrata:rowID>
    <rdfs:member
rdf:resource="https://data.cityofnewyork.us/resource/wng2-
85mv"/>
    <ds:id>1</ds:id>
    <ds:gis_id>15540</ds:gis_id>
    <ds:roadway_name>BEACH STREET</ds:roadway_name>
    <ds:from>UNION PLACE</ds:from>
    <ds:to>VAN DUZER STREET</ds:to>
    <ds:direction>NB</ds:direction>
    <ds:date>2012-01-09T00:00:00</ds:date>
    <ds:_12_00_1_00_am>20.00</ds:_12_00_1_00_am>
    <ds:_1_00_2_00am>10.00</ds:_1_00_2_00am>
    <ds:_2_00_3_00am>11.00</ds:_2_00_3_00am>

<ds:_11_00_12_00am>42.00</ds:_11_00_12_00am></rdf:Descriptio
n>
```

Figure 7-2 Sample of RDF/XML Traffic Volume Count Dataset

Association rule result in descending order Min_support =0.1, min_ confidence = 0.1

Association rules for Traffic Volume Dataset

Table 7-7 ARM for Traffic Volume Count Dataset

	antecedent	consequent	confidence
0	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_7_00_8_00pm', '6')]	0.79
1	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_4_00_5_00pm', '6')]	0.86
2	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_5_00_6_00pm', '6')]	0.84
3	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_11_00_12_00am', '4')]	0.89
4	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_3_00_4_00pm', '6')]	0.75
5	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_11_00_12_00pm', '6')]	0.93
6	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_1_00_2_00pm', '6')]	0.92
7	[('_10_00_11_00am', '6'), ('_6_00_7_00pm', '6'), ('_2_00_3_00pm', '6'), ('_10_00_11_00pm', '4')]	[('_9_00_10_00am', '6')]	0.83

Number of rules = 40586

All above rules do not give us full meaning about dataset, such as the direction of trip or street name. In order to extract rules that contain direction, we use SQL queries.

All items consist of predicate and object, the numbers in objects refer to number of vehicles. We group numbers to categories from 1 to 10 as follow

Table 7-8 Convert continuous numbers to categories

Traffic count	Category id
[0,15]	1
[15,30]	2
[30,60]	3
[60,100]	4
[100,250]	5
[250,300]	6
[300,600]	7
[600,800]	8
[800,1000]	9
[1000,1500]	10

The dataset contains continuous numbers, however, the ARM cannot work properly on continuous numbers; because this will generate large number of frequent pattern when minimum support is very low, in contrast and when minimum support is large, there will be few association rule or none number of rules, in order to avoid this problem we group numbers and convert continuous data to categorical data by using **Bucketizer backage** in *pyspark.ml.feature* [7] .This is a ready model in Apache Spark, it is used to group continuous data to groups, each group has unique number, for example, we group counts of traffic to ten categories, each category has one ID number as shown in following **Code 7-4**.

```
from pyspark.ml.feature import Bucketizer

bucketizer =
Bucketizer(splits=[0,15,30,60,100,250,300,600,800,1000,1500,
float('Inf') ],inputCol="values" ,outputCol="o")
```

Code 7-4 Convert continuous numeric data to categories

[7] https://spark.apache.org/docs/2.2.0/ml-features.html#bucketizer

In above code numbers between 0 and 15 are represented as 1, also numbers from 15 to 30 are represented as 2 and so on.

All above rules may not give us full meaning about dataset such as road name or direction, therefore if we want to display all association rules that contain the item (direction, NB) we use SQL queries.

7.2.5 Queries

Query1 display all association rule where consequent is (direction, NB)

Table 7-9 ARM for Traffic volume when direction NB

	antecedent	consequent	confidence
32	[('_5_00_6_00pm', '6'), ('_3_00_4_00pm', '6'), ('_4_00_5_00pm', '6')]	[('direction', 'NB')]	0.35
2780	[('_6_00_7_00pm', '6'), ('_5_00_6_00pm', '6')]	[('direction', 'NB')]	0.37
3015	[('_9_00_10_00pm', '4')]	[('direction', 'NB')]	0.31
3256	[('_11_00_12_00am', '4'), ('_10_00_11_00pm', '4')]	[('direction', 'NB')]	0.31
3687	[('_1_00_2_00am', '2')]	[('direction', 'NB')]	0.29
7386	[('_10_00_11_00pm', '4'), ('_9_00_10_00pm', '4')]	[('direction', 'NB')]	0.33

Query2 display all association rule where antecedent contains (direction, NB)

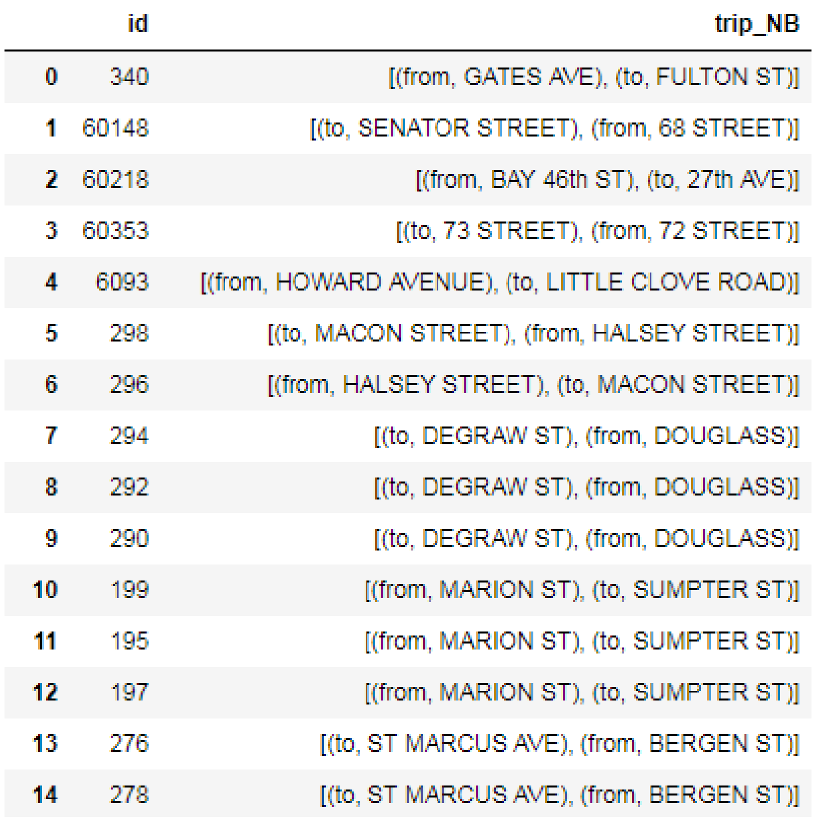
Table 7-10 Sub-model for Traffic ARM when antecedent is Direction NB

	antecedent	consequent	confidence
281	[('direction', 'NB'), ('_4_00_5_00pm', '6'), ('_9_00_10_00pm', '4')]	[('_10_00_11_00pm', '4')]	0.92
759	[('direction', 'NB'), ('_11_00_12_00am', '4')]	[('_10_00_11_00pm', '4')]	0.89
923	[('direction', 'NB'), ('_5_00_6_00pm', '6'), ('_3_00_4_00pm', '6')]	[('_4_00_5_00pm', '6')]	1
3840	[('direction', 'NB'), ('_10_00_11_00pm', '4')]	[('_8_00_9_00pm', '4')]	0.65
3841	[('direction', 'NB'), ('_10_00_11_00pm', '4')]	[('_4_00_5_00pm', '6')]	0.76
3842	[('direction', 'NB'), ('_10_00_11_00pm', '4')]	[('_6_00_7_00pm', '6')]	0.71

Number of rules 80

Filter all trips at NB direction

Table 7-11 Trips in NB direction



	id	trip_NB
0	340	[(from, GATES AVE), (to, FULTON ST)]
1	60148	[(to, SENATOR STREET), (from, 68 STREET)]
2	60218	[(from, BAY 46th ST), (to, 27th AVE)]
3	60353	[(to, 73 STREET), (from, 72 STREET)]
4	6093	[(from, HOWARD AVENUE), (to, LITTLE CLOVE ROAD)]
5	298	[(to, MACON STREET), (from, HALSEY STREET)]
6	296	[(from, HALSEY STREET), (to, MACON STREET)]
7	294	[(to, DEGRAW ST), (from, DOUGLASS)]
8	292	[(to, DEGRAW ST), (from, DOUGLASS)]
9	290	[(to, DEGRAW ST), (from, DOUGLASS)]
10	199	[(from, MARION ST), (to, SUMPTER ST)]
11	195	[(from, MARION ST), (to, SUMPTER ST)]
12	197	[(from, MARION ST), (to, SUMPTER ST)]
13	276	[(to, ST MARCUS AVE), (from, BERGEN ST)]
14	278	[(to, ST MARCUS AVE), (from, BERGEN ST)]

Number of trips in NB is 296

Above data frame refers to first fifteen trips at NB direction, therefore by matching those trips to the association rule result at NB direction we can understand human behavior in New York City at NB direction.

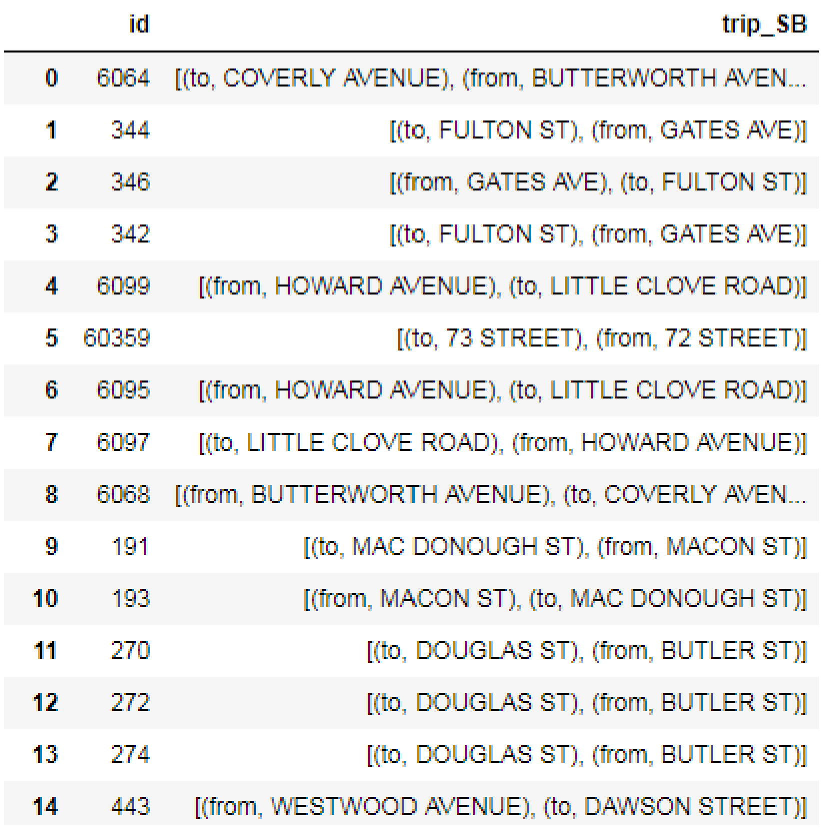
For example, if we take the rule of number 281 in the **Table 7-10** we can explain it as following expression **R**281 = [**IF** (direction, NB), (from, MARION ST), (to, SUMPTER ST), (4:00_5:00pm, 6), (9:00_10:00pm,4) **THEN** (10:00_11:00pm, 4).

Also we can express same rule by using another trip such as (from 72 street to 73 street)

Query2 display all association rule where antecedent contains (direction, SB)

First, we filter all trips in SB direction from dataset

Table 7-12 the trips in SB direction



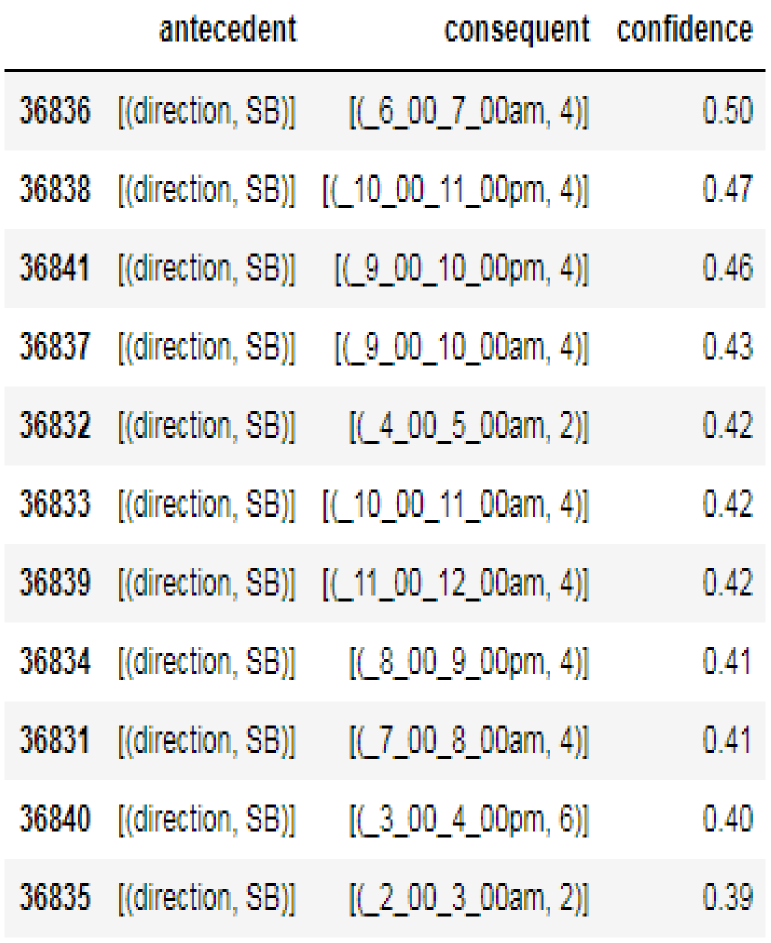
	id	trip_SB
0	6064	[(to, COVERLY AVENUE), (from, BUTTERWORTH AVEN...
1	344	[(to, FULTON ST), (from, GATES AVE)]
2	346	[(from, GATES AVE), (to, FULTON ST)]
3	342	[(to, FULTON ST), (from, GATES AVE)]
4	6099	[(from, HOWARD AVENUE), (to, LITTLE CLOVE ROAD)]
5	60359	[(to, 73 STREET), (from, 72 STREET)]
6	6095	[(from, HOWARD AVENUE), (to, LITTLE CLOVE ROAD)]
7	6097	[(to, LITTLE CLOVE ROAD), (from, HOWARD AVENUE)]
8	6068	[(from, BUTTERWORTH AVENUE), (to, COVERLY AVEN...
9	191	[(to, MAC DONOUGH ST), (from, MACON ST)]
10	193	[(from, MACON ST), (to, MAC DONOUGH ST)]
11	270	[(to, DOUGLAS ST), (from, BUTLER ST)]
12	272	[(to, DOUGLAS ST), (from, BUTLER ST)]
13	274	[(to, DOUGLAS ST), (from, BUTLER ST)]
14	443	[(from, WESTWOOD AVENUE), (to, DAWSON STREET)]

Number of trips 259.

All above trips at SB, we will match all those trips to generated association rule at SB direction. This will give us full meaning about traffic movement in each street.

Association rule at SB direction

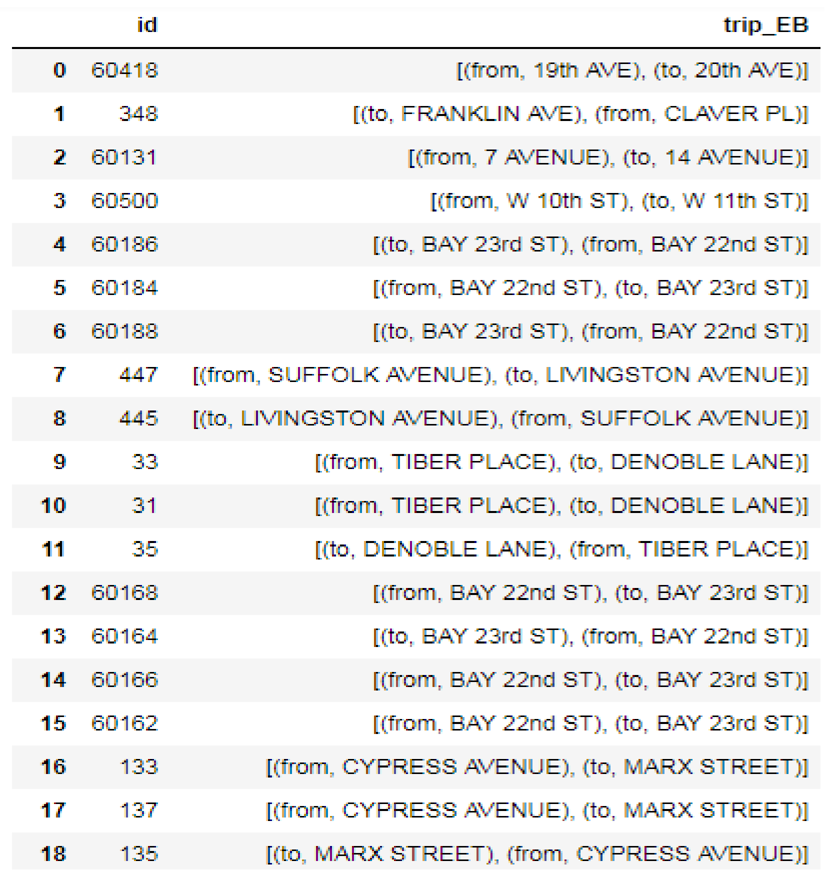
Table 7-13 ARM for Traffic volume when direction is SB



	antecedent	consequent	confidence
36836	[(direction, SB)]	[(_6_00_7_00am, 4)]	0.50
36838	[(direction, SB)]	[(_10_00_11_00pm, 4)]	0.47
36841	[(direction, SB)]	[(_9_00_10_00pm, 4)]	0.46
36837	[(direction, SB)]	[(_9_00_10_00am, 4)]	0.43
36832	[(direction, SB)]	[(_4_00_5_00am, 2)]	0.42
36833	[(direction, SB)]	[(_10_00_11_00am, 4)]	0.42
36839	[(direction, SB)]	[(_11_00_12_00am, 4)]	0.42
36834	[(direction, SB)]	[(_8_00_9_00pm, 4)]	0.41
36831	[(direction, SB)]	[(_7_00_8_00am, 4)]	0.41
36840	[(direction, SB)]	[(_3_00_4_00pm, 6)]	0.40
36835	[(direction, SB)]	[(_2_00_3_00am, 2)]	0.39

Query3 display all association rule where antecedent contains (direction, EB)

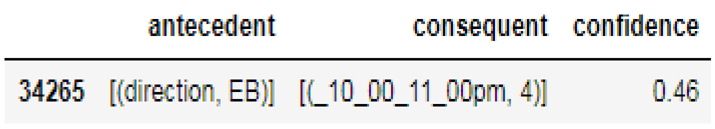
Table 7-14 the trips in EB direction



	id	trip_EB
0	60418	[(from, 19th AVE), (to, 20th AVE)]
1	348	[(to, FRANKLIN AVE), (from, CLAVER PL)]
2	60131	[(from, 7 AVENUE), (to, 14 AVENUE)]
3	60500	[(from, W 10th ST), (to, W 11th ST)]
4	60186	[(to, BAY 23rd ST), (from, BAY 22nd ST)]
5	60184	[(from, BAY 22nd ST), (to, BAY 23rd ST)]
6	60188	[(to, BAY 23rd ST), (from, BAY 22nd ST)]
7	447	[(from, SUFFOLK AVENUE), (to, LIVINGSTON AVENUE)]
8	445	[(to, LIVINGSTON AVENUE), (from, SUFFOLK AVENUE)]
9	33	[(from, TIBER PLACE), (to, DENOBLE LANE)]
10	31	[(from, TIBER PLACE), (to, DENOBLE LANE)]
11	35	[(to, DENOBLE LANE), (from, TIBER PLACE)]
12	60168	[(from, BAY 22nd ST), (to, BAY 23rd ST)]
13	60164	[(to, BAY 23rd ST), (from, BAY 22nd ST)]
14	60166	[(from, BAY 22nd ST), (to, BAY 23rd ST)]
15	60162	[(from, BAY 22nd ST), (to, BAY 23rd ST)]
16	133	[(from, CYPRESS AVENUE), (to, MARX STREET)]
17	137	[(from, CYPRESS AVENUE), (to, MARX STREET)]
18	135	[(to, MARX STREET), (from, CYPRESS AVENUE)]

Association rule at EB direction

Table 7-15 ARM when direction is EB



	antecedent	consequent	confidence
34265	[(direction, EB)]	[(_10_00_11_00pm, 4)]	0.46

By matching this rule to EB trips, above rule will be as R34265= [IF (direction, EB), (from, BAY 22nd ST), (to, BAY 23nd ST) **THEN** (10:00_11:00pm, 4)].

This rule mean "most of trips from BAY 22nd ST to BAY 23nd ST at EB direction, number of vehicles is about 80 at 10:00_11:00pm"

Query 4 display all association rules for (direction, WB)

First, filter all trips at WB direction

Table 7-16 The trips in WB direction

	id	trip_WB
0	60140	[(to, 14 AVENUE), (from, 7 AVENUE)]
1	60212	[(from, 19th AVE), (to, BAY 20th ST)]
2	60210	[(to, BAY 20th ST), (from, 19th AVE)]
3	60216	[(to, BAY 20th ST), (from, 19th AVE)]
4	60214	[(from, 19th AVE), (to, BAY 20th ST)]
5	60137	[(to, 14 AVENUE), (from, 7 AVENUE)]
6	60135	[(from, 7 AVENUE), (to, 14 AVENUE)]
7	60182	[(to, BAY 23rd ST), (from, BAY 22nd ST)]
8	60180	[(from, BAY 22nd ST), (to, BAY 23rd ST)]
9	449	[(to, LIVINGSTON AVENUE), (from, SUFFOLK AVENUE)]
10	39	[(to, DENOBLE LANE), (from, TIBER PLACE)]
11	37	[(to, DENOBLE LANE), (from, TIBER PLACE)]
12	6077	[(to, RENWICK AVENUE), (from, CAYUGA AVENUE)]
13	6075	[(from, CAYUGA AVENUE), (to, RENWICK AVENUE)]
14	432	[(from, MULBERRY AVENUE), (to, PARK DRIVE NORTH)]
15	430	[(to, PARK DRIVE NORTH), (from, MULBERRY AVENUE)]

Result of association rules at WB direction

Table 7-17 ARM when direction is WB

	antecedent	consequent	confidence
18285	[(direction, WB)]	[(_9_00_10_00pm, 4)]	0.56
18284	[(direction, WB)]	[(_10_00_11_00pm, 4)]	0.47

We can explain **R**18285 as "most of trips at WB direction from 7 AVENUE to 14 AVENUE at 9:00_10:00pm, the number of vehicles are about 80"

7.2.6 Advantages of using ARM on traffic volume count dataset

There are many advantage of mining traffic dataset, we can introduce some of these benefits as follow:

1. Determine relationships between numbers of traffic at each hour in each direction and street, this will help people to set traffic light on each hour and manage the NY city.
2. Enable us to understand human behavior in the city, besides this helping in a city planning by knowing transition trips and understand the most crowded road at each hour, in this field ARM has been used for shopping behavior (Yoshimura, Sobolevsky, Bautista Hobin, Ratti, & Blat, 2016) by extracting frequent trips between supermarkets for people.
3. Analyze traffic pattern and know peak hours at each street, as a result the streets that have the highest number of traffic at specific hour, they will require strong monitoring and management at peak hour

7.2.7 Metal Content of Consumer Products Dataset

This dataset is collected by the NYC Health Department called Department of Health and Mental Hygiene (DOHMH), this data shows the metal contents on each product and concentration of metal, also the country that produces this product

Number of triples: graph has 6500 statements.

Number of frequent columns: 3000.

Sample of dataset

```
<dsbase:da9u-wz3r
rdf:about="https://data.cityofnewyork.us/resource/da9u-
wz3r/1">
    <socrata:rowID>1</socrata:rowID>
    <rdfs:member
rdf:resource="https://data.cityofnewyork.us/resource/da
9u-wz3r"/>
    <ds:row_id>766</ds:row_id>
    <ds:product_type>Food-Spice</ds:product_type>
    <ds:product_name>Dry Mango Powder</ds:product_name>
    <ds:metal>Lead</ds:metal>
    <ds:concentration>-1</ds:concentration>
    <ds:units>ppm</ds:units>
    <ds:manufacturer>UNKNOWN           OR          NOT
STATED</ds:manufacturer>
    <ds:made_in_country>INDIA</ds:made_in_country>
    <ds:collection_date>2013-03-
29T00:00:00</ds:collection_date>
  <ds:deleted>No</ds:deleted></dsbase:da9u-wz3r>
```

Figure 7-3 RDF/XML Metal Content dataset

Minimum Support=0.01, Minimum Confidence=0.02

Association rules between items

Table 7-18 ARM for metal content

	antecedent	consequent	confidence
1	[('product_type', 'Food-Spice')]	[('metal', 'Lead')]	1
2	[('product_type', 'Food-Spice'), ('concentration', '-1')]	[('metal', 'Lead')]	1
15	[('product_type', "Toys/Children's Products")]	[('metal', 'Lead')]	1
27	[('made_in_country', 'UNKNOWN OR NOT STATED')]	[('metal', 'Lead')]	0.91
14	[('made_in_country', 'UNKNOWN OR NOT STATED'), ('concentration', '-1')]	[('metal', 'Lead')]	0.9
3	[('metal', 'Mercury')]	[('product_type', 'Dietary Supplement/Medications/Remedy')]	0.82
18	[('concentration', '-1')]	[('metal', 'Lead')]	0.81

7.2.8 Queries

Query1 if antecedent = (metal, lead) then consequent =?

Result

Table 7-19 ARM for metal content when antecedent contains lead metal

	antecedent	consequent	confidence
202	[('product_name', 'Turmeric powder'), ('metal', 'Lead')]	[('product_type', 'Food-Spice')]	1
215	[('product_name', 'Georgian spice'), ('made_in_country', 'GEORGIA'), ('metal', 'Lead')]	[('product_type', 'Food-Spice')]	1
194	[('product_name', 'Turmeric'), ('metal', 'Lead')]	[('product_type', 'Food-Spice')]	1
111	[('product_name', 'Chili powder'), ('made_in_country', 'BANGLADESH'), ('metal', 'Lead')]	[('product_type', 'Food-Spice')]	1
187	[('product_name', 'Chili powder'), ('metal', 'Lead')]	[('product_type', 'Food-Spice')]	1
349	[('product_type', 'Food-Candy'), ('made_in_country', 'UNKNOWN OR NOT STATED'), ('metal', 'Lead')]	[('concentration', '-1')]	1

Query2: IF antecedent = (made In, China) or (made In, Lebanon) THEN?

Result

Table 7-20 ARM for products that made in China AND Lebanon

	antecedent	consequent	confidence
22	[('product_name', 'Crayon'), ('made_in_country', 'CHINA')]	[('product_type', "Toys/Children's Products")]	1
20	[('product_name', "Emperor's Tea Pill (Conc)/Tian Wang Bu Xin Wan"), ('made_in_country', 'CHINA')]	[('product_type', 'Dietary Supplement/Medications/Remedy')]	1
17	[('made_in_country', 'LEBANON')]	[('product_type', 'Cosmetics')]	1
21	[('made_in_country', 'LEBANON'), ('metal', 'Mercury'), ('concentration', '-1')]	[('product_type', 'Cosmetics')]	1
23	[('product_name', 'Crayon'), ('made_in_country', 'CHINA')]	[('metal', 'Lead')]	1
26	[('metal', 'Arsenic'), ('made_in_country', 'CHINA')]	[('product_type', 'Dietary Supplement/Medications/Remedy')]	1

Number of rules is 45

7.3 CityPulse dataset

CityPulse website [8] contains different annotated datasets for smart city applications such as pollution, traffic, and weather dataset, in this research we used two types of dataset, the primary goal to use ARM in this data is improving RDF graph and summarize it.

[8] http://iot.ee.surrey.ac.uk:8080/datasets.html

7.3.1 Cultural Events Dataset

This dataset consists of series of events that happened in city of Aarhus in Denmark, this data is collected in period of May 5th 2014 - January 25th 2015, although these events seem not clear for us, but extract frequent events will help people in Denmark to understand historical events and use this knowledge in a near future.

Data format: turtle data

Download site:

http://iot.ee.surrey.ac.uk:8080/datasets/aarhusculturalevents/culturalEvents_aarhus.ttl

Dataset name: Cultural Event data

Number of triples: graph has 1391 statements.

Volume of data: 100,732 bytes

Number of frequent columns: 99

Sample of dataset

```
<http://iot.ee.surrey.ac.uk/citypulse/datasets/aarhuscu
lturalevents/culturalEvents_aarhus#observations_point_k
28nge120jv0o1q36pt28g9tf7> a sao:Point ;
       sao:hasUnitOfMeasurement unit0:eventData ;
       ns1:featureOfInterest
<http://iot.ee.surrey.ac.uk/citypulse/datasets/aarhuscu
lturalevents/culturalEvents_aarhus#context_slllpqckbsul
250sfuqrg9qo31> ;
       sao:time [a tl:Instant;
              tl:at "2014-05-03T00:00:00"^^xsd:dateTime
];
       sao:value "Title: SPOT Festival 2014 Price:
Billet URL:
http://www.billetlugen.dk/referer/?r=266abe1b7fab4562a5
c2531d0ae62171&p=/koeb/billetter/33768/47633/" .
```

Figure 7-4 TURTLE Culture Event Dataset

The results of mining object where predicate is target

	antecedent	consequent	confidence
0	['Title: Tam Tam i Musikhuset Price: 275.00 - 485.00 DKK ', 'Title: Hist & Her - Sanseudstilling Price: Gratis ']	['Title: 8000 Comedy Price: Gratis ']	1
1	['Title: Familiekoncert - gratis Price: Gratis ']	['Title: 8000 Comedy Price: Gratis ']	1
5	['Title: Tam Tam i Musikhuset Price: 275.00 - 485.00 DKK ']	['Title: Hist & Her - Sanseudstilling Price: Gratis ']	1
6	['Title: Tam Tam i Musikhuset Price: 275.00 - 485.00 DKK ']	['Title: 8000 Comedy Price: Gratis ']	1
7	['Title: Hist & Her - Sanseudstilling Price: Gratis ', u'Title: 8000 Comedy Price: Gratis ']	['Title: Tam Tam i Musikhuset Price: 275.00 - 485.00 DKK ']	1
8	['Title: Tam Tam i Musikhuset Price: 275.00 - 485.00 DKK ','Title: 8000 Comedy Price: Gratis ']	['Title: Hist & Her - Sanseudstilling Price: Gratis ']	1
9	['Title: Hist & Her - Sanseudstilling Price: Gratis ']	['Title: Tam Tam i Musikhuset Price: 275.00 - 485.00 DKK ']	1
10	['Title: Hist & Her - Sanseudstilling Price: Gratis ']	['Title: 8000 Comedy Price: Gratis ']	1
2	['Title: 8000 Comedy Price: Gratis ']	['Title: Familiekoncert - gratis Price: Gratis ']	0.5
3	['Title: 8000 Comedy Price: Gratis ']	['Title: Tam Tam i Musikhuset Price: 275.00 - 485.00 DKK ']	0.5
4	['Title: 8000 Comedy Price: Gratis ']	['Title: Hist & Her - Sanseudstilling Price: Gratis ']	0.5

Table 7-21 ARM for Culture Events Dataset

7.3.2 Road Traffic Dataset

Vehicle traffic dataset is collected in Denmark, it consists of 449 observations that are taken by sensors in the period of six months.

We extract frequent numbers of vehicle that occurred at same hour, this method is used to improve RDF graph (Abedjan & Naumann, 2013) and extract frequent pattern from big graph.

Data format: turtle data

Download site: http://iot.ee.surrey.ac.uk:8080/datasets.html#traffic

Dataset name: Traffic data

Number of triples: graph has 340961 statements.

Volume of data: 10.4 GB

Number of frequent columns: 37885

Minimum Support=0.2

Minimum Confidence=0.3

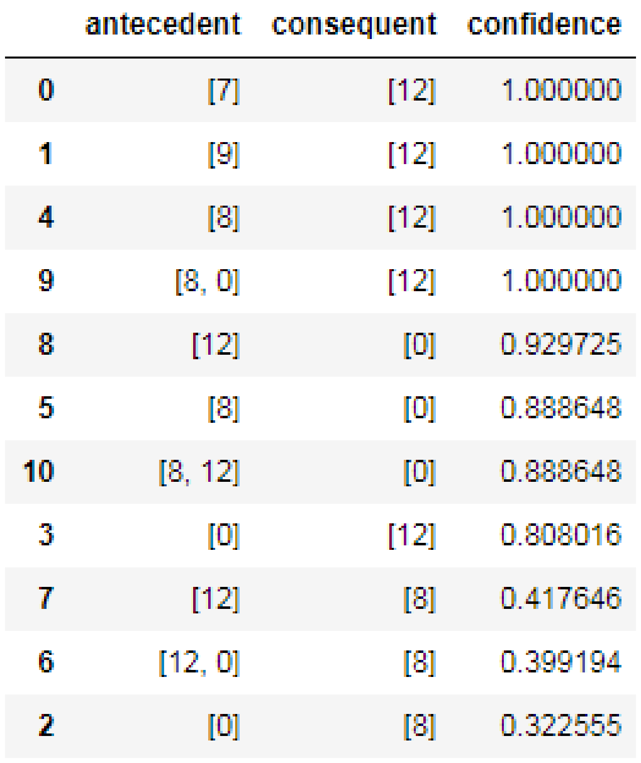
Before applying ARM on this data, we grouped continuous data to 16 categories, these categories represent number of traffic at two points in Aarhus city in Denmark.

For example we give the numbers between 0 and 30 one, and the numbers between 30 and 60 two and so on.

We also group numbers that occur at same hour and date in one transaction, because dataset give us full description about date and time.

Association rules

Table 7-22 values that occur in the same hour for Road Traffic Dataset



	antecedent	consequent	confidence
0	[7]	[12]	1.000000
1	[9]	[12]	1.000000
4	[8]	[12]	1.000000
9	[8, 0]	[12]	1.000000
8	[12]	[0]	0.929725
5	[8]	[0]	0.888648
10	[8, 12]	[0]	0.888648
3	[0]	[12]	0.808016
7	[12]	[8]	0.417646
6	[12, 0]	[8]	0.399194
2	[0]	[8]	0.322555

Number of rules: only 10 rules

All above rules represent traffic that occur together at same time, however number of observations are about 449, only ten of them are frequent pattern, this enable us to summarize big graph and prevent the problem of explosion storage and discover range of vehicles in the city. In above ARM, we notice that number of vehicles between 45 and 100, the rule 0 means “if number of vehicles is about [45:50] the next number of vehicle at next hour will be [70:100]” with confidence 100% while all numbers in above ARM refer to category number of vehicle as shown in table 7-22.

Table 7-23 Group number of vehicles

Vehicle count	**Category id**
0-15	0
15-20	1
20-25	2
25-30	3
30-35	4
35-40	5
40-45	6
45-50	7
50-55	8
55-60	9
60-65	10
65-70	11
70-100	12
100-200	13
200-250	14
250-300	15

7.3.3 Pollution Measurements for the City of Brasov in Romania

A collection of pollution measurements designed to complement the vehicle traffic dataset above. They simulated one sensor for each of the traffic sensor at the exact location of this traffic sensor.

The pollution sensors measure air quality in each location such as ozone, particulate matter, carbon monoxide, sulfur dioxide, and nitrogen dioxide. We mined carbon monoxide values that occurred at same day

Data format: turtle data

Download site: http://iot.ee.surrey.ac.uk:8080/datasets.html#pollution

Dataset name: Pollution data

Number of triples: graph has 340961 statements.

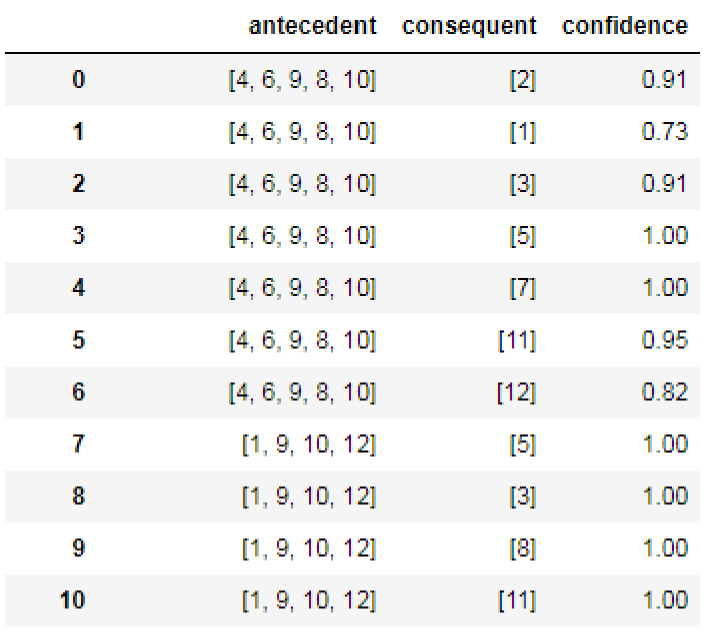
Volume of data: 10.4 GB

Number of frequent columns: 245957

Minimum Support=0.2, Minimum Confidence=0.3

Association rules

Table 7-24 ARM for Pollution Dataset



	antecedent	consequent	confidence
0	[4, 6, 9, 8, 10]	[2]	0.91
1	[4, 6, 9, 8, 10]	[1]	0.73
2	[4, 6, 9, 8, 10]	[3]	0.91
3	[4, 6, 9, 8, 10]	[5]	1.00
4	[4, 6, 9, 8, 10]	[7]	1.00
5	[4, 6, 9, 8, 10]	[11]	0.95
6	[4, 6, 9, 8, 10]	[12]	0.82
7	[1, 9, 10, 12]	[5]	1.00
8	[1, 9, 10, 12]	[3]	1.00
9	[1, 9, 10, 12]	[8]	1.00
10	[1, 9, 10, 12]	[11]	1.00

Number of rules 24610

7.3.4 Advantage of using ARM on Road Traffic & pollution dataset

- Improve RDF graph by storing frequent values only.
- Summarize RDF graph.
- Avoid explosion storage.
- Discover range of number of vehicles in the city.

7.4 Kaggle dataset

Kaggle website is the most recommended website, it contains many machine learning projects and datasets in various applications. We choose two datasets from this website.

7.4.1 Smart meters in Landon

Smart meters dataset is downloaded from Kaggle website[9], it is stored as CSV file, we converted this dataset to semantic data then we applied our ARM on it. This dataset describes weather information in London such as temperature, visibility, and pressure, all data is collected by smart sensors therefore it is a good example for IOT applications.

Dataset name: weather hourly dark sky dataset

Type: CSV

Link to dataset: https://www.kaggle.com/jeanmidev/smart-meters-in-london/data

How it collected: by sensors in London city

Sample of data

[9] https://www.kaggle.com/datasets

Table 7-25 Sample of smart meters dataset

visibility	windBeari	temperatu	time	dewPoint	pressure	apparentT	windSpee	precipTyp	icon	humidity	summary
5.97	104	10.24	########	8.86	1016.76	10.24	2.77	rain	partly-clo	0.91	Partly Cloudy
4.88	99	9.76	########	8.83	1016.63	8.24	2.95	rain	partly-clo	0.94	Partly Cloudy
3.7	98	9.46	########	8.79	1016.36	7.76	3.17	rain	partly-clo	0.96	Partly Cloudy
3.12	99	9.23	########	8.63	1016.28	7.44	3.25	rain	fog	0.96	Foggy
1.85	111	9.26	########	9.21	1015.98	7.24	3.7	rain	fog	1	Foggy
1.96	115	9.33	########	8.87	1015.91	7.19	3.97	rain	fog	0.97	Foggy
1.3	118	9.31	########	8.82	1015.7	7.1	4.1	rain	fog	0.97	Foggy
1.22	114	8.85	########	8.69	1016.08	6.48	4.23	rain	fog	0.99	Foggy
1.4	120	9.13	########	8.75	1016.33	6.84	4.2	rain	fog	0.97	Foggy
1.38	121	9.23	########	8.7	1016.57	7.07	3.96	rain	fog	0.97	Foggy
1.35	115	9.21	########	8.76	1016.26	6.96	4.16	rain	fog	0.97	Foggy
1.72	127	9.78	########	9.23	1016.17	7.68	4.14	rain	fog	0.96	Foggy
1.83	129	9.91	########	9.34	1015.92	7.91	3.97	rain	fog	0.96	Foggy
2.53	120	10.22	########	9.73	1015.49	10.22	4.25	rain	fog	0.97	Foggy
3.67	122	10.63	########	9.73	1015.1	10.63	4.28	rain	partly-clo	0.94	Mostly Cloudy
3.77	126	10.34	########	9.7	1015.32	10.34	4.3	rain	partly-clo	0.96	Mostly Cloudy

In order to convert this data to semantic data, we add id column to it, the id can be considered as URI for sensor, however there are ten types of sensors here, therefor we can say this URI works as Bnode, this Bnode combines between two things (type and date) as following graph:

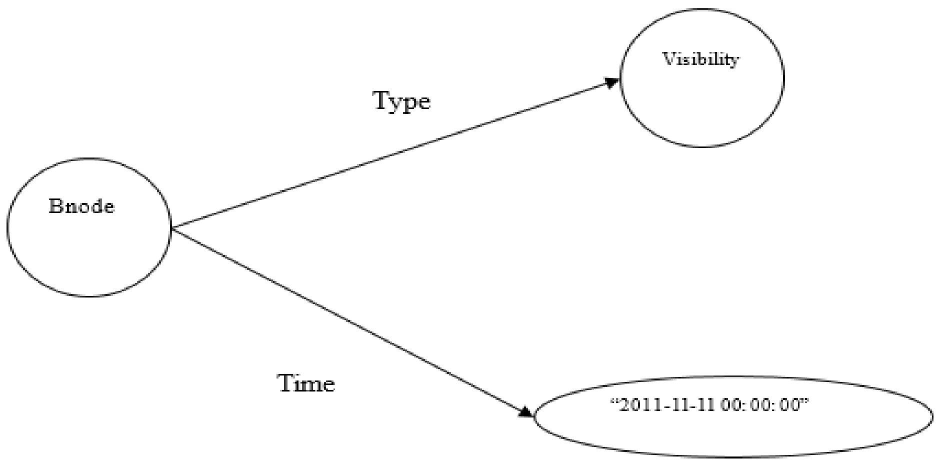


Figure 7-5 Bnode as subject

In ARM, we combine between items that *occur together*, therefore by ignoring type of sensor we *combine by date* all values that occur in the same time. This type of graph is similar to CityPulse TURTLE graphs, where Bnode combines between time and type . In this sematic graph, we also assigned values that occur in same time into unique key.

Semantic data means triples, in below table we convert this data to triples

Table 7-26 RDF graph for weather dataset

subject	predicate	object
:1	:visibility	5.97
:1	:windBeering	104
:1	:temperature	10.24
:1	:dewPoint	8.86
:1	:pressure	1016.78

After convert data to triples, we apply our algorithm directly on it.

Association rules

Table 7-27 ARM for weather dataset

	antecedent	consequent	confidence
0	[('apparentTemperature', '11'), ('temperature', '11'), ('precipType', ' rain')]	[('icon', ' partly-cloudy-day')]	0.37
1	[('apparentTemperature', '11'), ('temperature', '11'), ('precipType', ' rain')]	[('pressure', '3')]	0.91
2	[('apparentTemperature', '11'), ('temperature', '11'), ('precipType', ' rain')]	[('dewPoint', '11')]	0.37
3	[('temperature', '11')]	[('apparentTemperature', '11')]	1
4	[('temperature', '11')]	[('dewPoint', '11')]	0.37
5	[('temperature', '11')]	[('icon', ' partly-cloudy-day')]	0.37
6	[('temperature', '11')]	[('pressure', '3')]	0.91
7	[('temperature', '11')]	[('precipType', ' rain')]	1
8	[('pressure', '3'), ('precipType', ' rain')]	[('apparentTemperature', '11')]	0.29
9	[('pressure', '3'), ('precipType', ' rain')]	[('windspeed', '1')]	0.13
10	[('pressure', '3'), ('precipType', ' rain')]	[('icon', ' partly-cloudy-day')]	0.38

Number of rules 307

7.4.1.1 ARM as classifier

If we set consequent to be only summary values, we will get classifier association rule, this classifier can give us the potential relation between attributes and classes as shown in below, we used SQL queries to extract the rules that their consequent is summary values.

Table 7-28 classifier ARM for weather dataset

	antecedent	consequent	confidence
0	[('apparentTemperature', '11'), ('temperature', '11'), ('precipType', ' rain')]	[('icon', ' partly-cloudy-day')]	0.37
1	[('apparentTemperature', '11'), ('temperature', '11'), ('precipType', ' rain')]	[('pressure', '3')]	0.91
2	[('apparentTemperature', '11'), ('temperature', '11'), ('precipType', ' rain')]	[('dewPoint', '11')]	0.37
3	[('temperature', '11')]	[('apparentTemperature', '11')]	1
4	[('temperature', '11')]	[('dewPoint', '11')]	0.37
5	[('temperature', '11')]	[('icon', ' partly-cloudy-day')]	0.37
6	[('temperature', '11')]	[('pressure', '3')]	0.91
7	[('temperature', '11')]	[('precipType', ' rain')]	1
8	[('pressure', '3'), ('precipType', ' rain')]	[('apparentTemperature', '11')]	0.29
9	[('pressure', '3'), ('precipType', ' rain')]	[('windspeed', '1')]	0.13
10	[('pressure', '3'), ('precipType', ' rain')]	[('icon', ' partly-cloudy-day')]	0.38

We notice that ARM can work as classifier at some cases, the researchers in (Atzmueller et al., 2018; Nguyen, Nguyen, Vo, & Pedrycz, 2016; Nithya & Duraiswamy, 2014) have used fuzzy association rules to classify diseases, for example in the above rules, we query the rules while consequent is only summary day, in this state ARM can work as classifier.

7.4.1.2 Benefit of association rules on weather hourly dark sky data

ARM enable us to detect relationships between different types of values such as temperature, pressure, and wind speed, however the rules of high confidence can be reused to detect missing values, for example when rule is if "(apparent Temperature, 11) & (icon, partlyCloudyDay) then (temperature, 11)" therefore if we do not know temperature, we can use this rule to infer it, besides ARM can be used as feature selection in various data mining algorithm.

7.4.2 Pima Indians Diabetes Database

This dataset is very common dataset, it is available in most dataset repositories websites, this dataset usually used for classification, in this study, we also use this data to make classification by using association rule mining.

In this research (Mezghani et al., 2015), the researchers have used smart wearable sensors to detected diabetes, for example the patient pressure can be collected by wearable pressure sensor that may be shoes or watch also the glucose can be measured by glucose lens and all these wearable sensors are connected to one cloud by internet as shown in picture below.

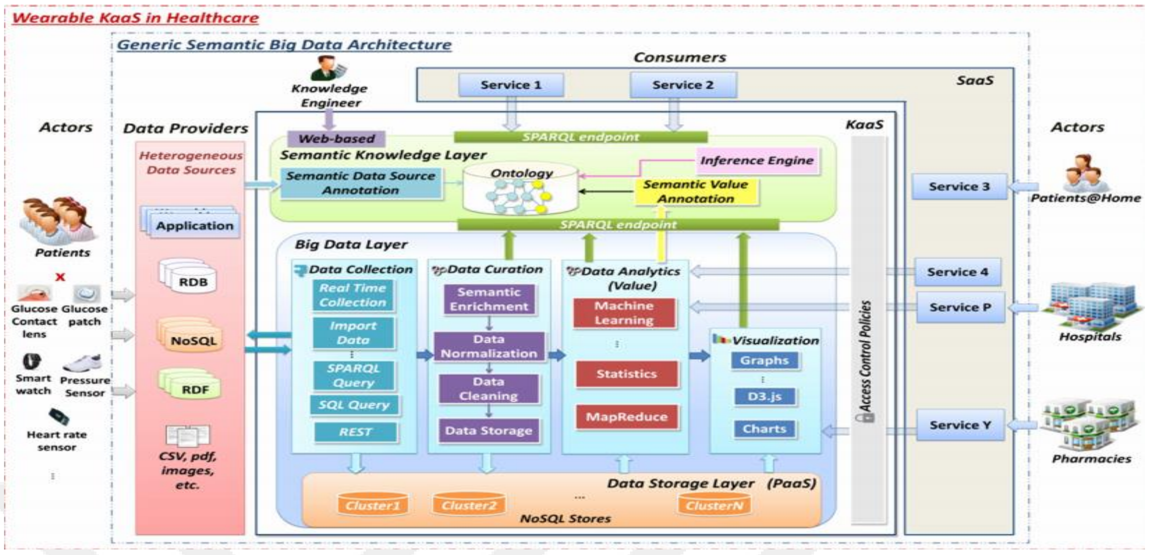


Figure 7-6 Connect wearable devices to cloud

Pima dataset contains patient information such as pressure, glucose, and insulin. By assuming that some of these values are collected by wearable sensors as above picture, this dataset will be an interesting example in the field of IoT in healthcare system.

In healthcare system, we need to combine between heterogeneous data for same patient for example type of drug, age, pressure, and insulin, all those information can be connected together by convert all data in hospital system to semantic data then we can connect all data to internet, also those information can be shared to another hospital system to detect the patient status for other patient easily a cross internet.

Table 7-29 Convert dataset to triples

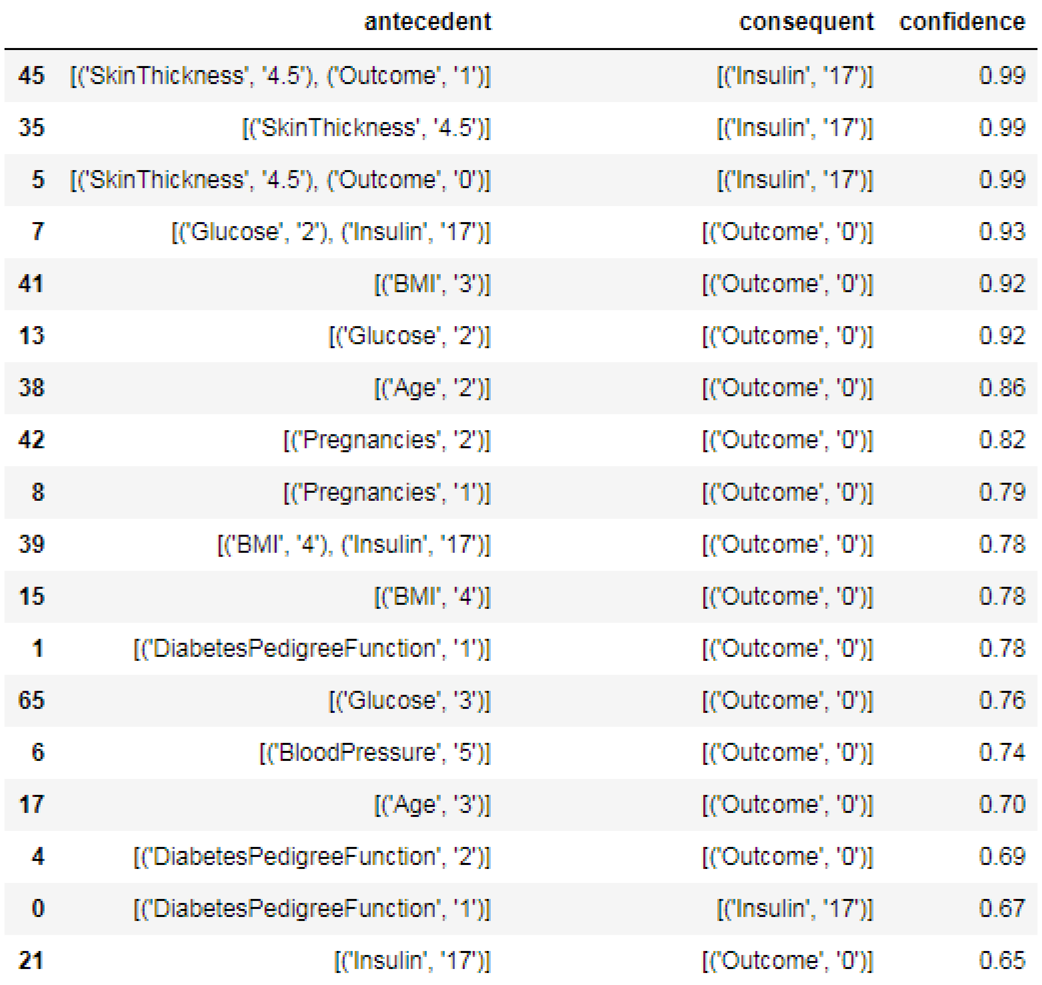
s	p	o
1	Pregnancies'	'6'
1	Age'	'8'
1	DiabetesPedigreeFunction'	'6'
1	BMI'	'5'
1	Insulin'	'17'
1	SkinThickness'	'6'
1	BloodPressure'	'7'
1	Glucose'	'7'
1	Outcome'	'1'

ARM on Pima dataset step by step

First, we convert data to triples where patient ID is subject, predicate is the name of a column such as blood pressure or insulin, and object is the value of attribute as shown in above **Table 7-28.** Second, we apply our algorithm on these triples. Third, generate association rules. Finally, classify data by using ARM.

7.4.2.1 ARM for Pima dataset

Table 7-30 ARM for diabetes dataset



	antecedent	consequent	confidence
45	[('SkinThickness', '4.5'), ('Outcome', '1')]	[('Insulin', '17')]	0.99
35	[('SkinThickness', '4.5')]	[('Insulin', '17')]	0.99
5	[('SkinThickness', '4.5'), ('Outcome', '0')]	[('Insulin', '17')]	0.99
7	[('Glucose', '2'), ('Insulin', '17')]	[('Outcome', '0')]	0.93
41	[('BMI', '3')]	[('Outcome', '0')]	0.92
13	[('Glucose', '2')]	[('Outcome', '0')]	0.92
38	[('Age', '2')]	[('Outcome', '0')]	0.86
42	[('Pregnancies', '2')]	[('Outcome', '0')]	0.82
8	[('Pregnancies', '1')]	[('Outcome', '0')]	0.79
39	[('BMI', '4'), ('Insulin', '17')]	[('Outcome', '0')]	0.78
15	[('BMI', '4')]	[('Outcome', '0')]	0.78
1	[('DiabetesPedigreeFunction', '1')]	[('Outcome', '0')]	0.78
65	[('Glucose', '3')]	[('Outcome', '0')]	0.76
6	[('BloodPressure', '5')]	[('Outcome', '0')]	0.74
17	[('Age', '3')]	[('Outcome', '0')]	0.70
4	[('DiabetesPedigreeFunction', '2')]	[('Outcome', '0')]	0.69
0	[('DiabetesPedigreeFunction', '1')]	[('Insulin', '17')]	0.67
21	[('Insulin', '17')]	[('Outcome', '0')]	0.65

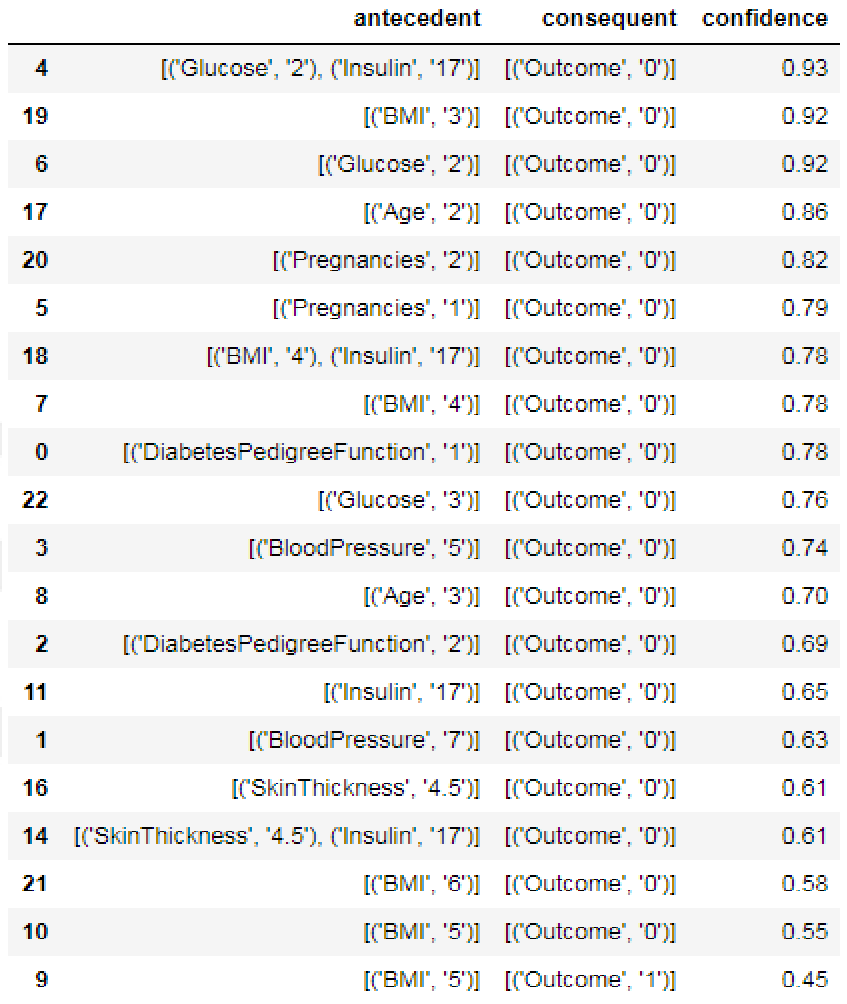
Number of rules =66

7.4.2.2 ARM as classifier

Display association rule in which the consequent is either (outcome, 0) or (outcome, 1)

Result of association rule in which the confidence is ordered in descending order

Table 7-31 ARM as classifier for diabetes dataset



	antecedent	consequent	confidence
4	[('Glucose', '2'), ('Insulin', '17')]	[('Outcome', '0')]	0.93
19	[('BMI', '3')]	[('Outcome', '0')]	0.92
6	[('Glucose', '2')]	[('Outcome', '0')]	0.92
17	[('Age', '2')]	[('Outcome', '0')]	0.86
20	[('Pregnancies', '2')]	[('Outcome', '0')]	0.82
5	[('Pregnancies', '1')]	[('Outcome', '0')]	0.79
18	[('BMI', '4'), ('Insulin', '17')]	[('Outcome', '0')]	0.78
7	[('BMI', '4')]	[('Outcome', '0')]	0.78
0	[('DiabetesPedigreeFunction', '1')]	[('Outcome', '0')]	0.78
22	[('Glucose', '3')]	[('Outcome', '0')]	0.76
3	[('BloodPressure', '5')]	[('Outcome', '0')]	0.74
8	[('Age', '3')]	[('Outcome', '0')]	0.70
2	[('DiabetesPedigreeFunction', '2')]	[('Outcome', '0')]	0.69
11	[('Insulin', '17')]	[('Outcome', '0')]	0.65
1	[('BloodPressure', '7')]	[('Outcome', '0')]	0.63
16	[('SkinThickness', '4.5')]	[('Outcome', '0')]	0.61
14	[('SkinThickness', '4.5'), ('Insulin', '17')]	[('Outcome', '0')]	0.61
21	[('BMI', '6')]	[('Outcome', '0')]	0.58
10	[('BMI', '5')]	[('Outcome', '0')]	0.55
9	[('BMI', '5')]	[('Outcome', '1')]	0.45

Number of rules = 23

7.4.3 Frequent two-factor set for Pima dataset by using SAG algorithm

Table 7-32 Frequent two factor set by SAG algorithm

	2-Itemset
0	(('Age', 5.0), ('Outcome', 0.0))
1	(('Age', 5.0), ('BloodPressure', 7.0))
2	(('Age', 5.0), ('BMI', 4.0))
3	(('Age', 5.0), ('Glucose', 2.0))
4	(('Age', 5.0), ('Insulin', 17.0))
5	(('Age', 5.0), ('BMI', 6.0))
6	(('Age', 5.0), ('SkinThickness', 4.5))
7	(('Age', 5.0), ('Pregnancies', 5.0))
8	(('Age', 5.0), ('BMI', 5.0))
9	(('Age', 5.0), ('SkinThickness', 5.0))
10	(('Age', 5.0), ('DiabetesPedigreeFunction', 2.0))
11	(('Age', 5.0), ('Outcome', 1.0))
12	(('Age', 7.0), ('Outcome', 0.0))
13	(('Age', 7.0), ('BloodPressure', 7.0))
14	(('Age', 7.0), ('Insulin', 17.0))
15	(('Age', 7.0), ('SkinThickness', 4.5))
16	(('Age', 7.0), ('BMI', 5.0))
17	(('Age', 7.0), ('Outcome', 1.0))
18	(('Age', 2.0), ('Outcome', 0.0))
19	(('Age', 2.0), ('BloodPressure', 4.0))
20	(('Age', 2.0), ('BMI', 7.0))
21	(('Age', 2.0), ('SkinThickness', 3.0))
22	(('Age', 2.0), ('BloodPressure', 7.0))
23	(('Age', 2.0), ('Insulin', 217.0))
24	(('Age', 2.0), ('Glucose', 5.0))
25	(('Age', 2.0), ('DiabetesPedigreeFunction', 4.0))
26	(('Age', 2.0), ('Insulin', 19.0))
27	(('Age', 2.0), ('BloodPressure', 6.0))
28	(('Age', 2.0), ('Pregnancies', 0.0))

Number of rules are 148

Target value is **Age**

Minimum support= 0.013

Support count = minimum support * number of transactions

Support count= 0.013 * 768=10

7.4.3.1 Advantages of association rules in Pima Diabetes Dataset

We can conclude advantages in three points

- Generate potential rules for each class
- Select important items which have highest confidence with each classes and ignore others with low confidence, also if the relation between two items is very high confidence then we can choose antecedent only or consequent only.
- We can use ARM model to detect missing values.

7.5 The execution time for PFP algorithm on cluster of nodes

In this part, we calculate the execution time for PFP algorithm on different size of semantic data and different number of nodes, first, we ordered data according on number of triples in ascending order

1. Cultural event dataset (1391 triples)
2. Metal content dataset (6500 triples)
3. Diabetes dataset (6913 triples)
4. PPTs dataset (17384 triples)
5. Traffic volume count (30460 triples)
6. Smart meters in London (232816 triples)
7. Pollution dataset (340961 triples)

We set number of nodes in each execution and we calculate the execution time in each number of node.

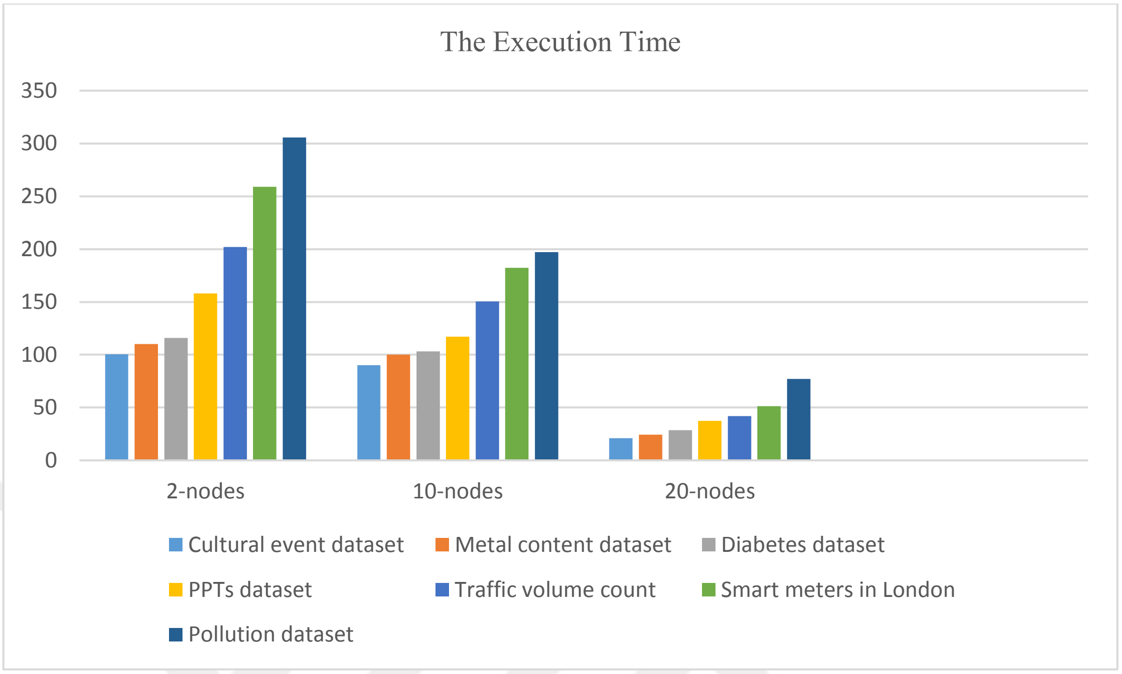


Figure 7-7 Performance of PFP algorithm on different size dataset

From above results we notice that execution time is decreasing when number of node increases. On the other hand there is a direct relation between time and size of data, but when the number of node is 20, there is no significant difference between execution time on different size dataset. Apache Spark is designed to process big data, but when dataset is small and it can be handled in one machine, it is the better to use one machine instead of cluster of nodes in Apache Spark. The author "Spark-in-Action" book says "if you don't have a large amount of data, Spark may not be required, because it needs to spend some time setting up jobs, tasks, and so on. Sometimes a simple relational database or a set of clever scripts can be used to process data more quickly than a distributed system such as Spark" (Zecevic & Bonaci, 2016).

7.6 Chapter summary

In this chapter, we explained dataset that used in this study, we gave full description for dataset such as number of triples, name, URL, and type of graph, after that we tested this dataset by using two methods, first method is mining RDF graph and extract frequent (predicate & object), we explained the advantages of using ARM on each dataset, however we have used different dataset from different domains such as traffic dataset, health dataset, while association rules can play different roles in each domain, also the applications that have same subject can integrate together and sharing knowledge, second method is used to summarize big RDF graph by extract frequent objects that occurred together in the same time, the result showed only few of objects are frequent from large numbers of triples.

We also tested scalability of PFP algorithm by adjusting the number of nodes in Apache Spark, the results show that execution time is decreasing when number of nodes increase.

In diabetes and weather datasets, we concluded that ARM can work as classifier in some cases, this can be done by extract the rules which consequent consists of state of patient in diabetes dataset and summary day in weather dataset.

DANIEL T. LAROSE has referred in his book (Larose & Larose, 2014) that association rule mining can represent supervised learning if it works as classifier, on the other hand ARM can be unsupervised algorithm when it extracts association rule from data.

CHPATER 8

8 Conclusion and Future work

8.1 Conclusion

In summary, we can see that Association Rule Mining (ARM) has a great effect on many IoT semantic data domains, including traffic, weather, and medicine, because applying ARM to semantic data can extract relations between items from different domains. In the hospital system, we can apply ARM to detect relations between different domains for the same patient, such as type of drug, type of disease, pressure, age and so on, because in semantic data, each patient has an identifier URI which is used as a subject. The predicate would be one of age, type of disease or gender. The URI can work as a primary key in relational databases (de Medeiros et al., 2015).

The integration of the RDF graph model technique and association rule mining is a challenging issue. First, the challenge is that semantic datasets are heterogeneous and they connect with many relations, while the datasets in tabular formats, such as market basket analysis, have only one relation, namely ‘buy.’ Second, the datasets in an RDF graph consist of triples and there is no exact definition for a transactions dataset. We handled these challenges by combing both (predicate and object) with the same subject. In traditional programs, combining many triples with the same key may occur using for or while loops. However, in Apache Spark, this occurs easily using the combineByKey method in a parallel fashion.

After we processed the datasets and transform them into a specific schema, such as [subject, [(p1,o1), (p2,o2),….]], we applied the PFP algorithm available in the Apache Spark machine learning APIs. The result of the association rules model shows different frequent predicates and objects from the semantic dataset.

In order to select a preferred association rule, we applied SQL queries to our model and extracted many sub-models from the original model. In special cases, we noted that ARM can work as a classifier, such as in the results of hourly weather and pima diabetes datasets. When we adjusted the consequent to contain only classes, we obtained some association rules that classify classes, but it was not a reliable classifier because the generated rules are potential statements and not logical. This type of classification is useful in patient diagnosis because the rules describe the most related attributes for each disease.

Association rule mining is usually used to select important attributes (Pei & Kamber, 2011). For example, if the correlation between two items is very high confidence, we will select antecedent items instead of all the data.

The RDF graph contains too much data and it is necessary to extract the important information from this large graph. Association rule mining is used to improve RDF data and extract frequent patterns only (Abedjan & Naumann, 2013) by using six types of configuration. In this study, we mined the object when the context was a predicate to discover a range for object values.

We conclude that association rule mining (ARM) is a supervised algorithm when it works as a classifier, including the Pima and weather datasets. Moreover, association rule mining (ARM) can be unsupervised learning when it extract rules between items.

Our algorithm can work properly in both categories of items, such as names, types and for continuous values such as temperature values.

The results show that execution time decreases when the number of nodes increases. On the other hand, there is a direct relation between time and the size of data.

8.2 Future work

In future investigations, it may be possible to apply association rules directly to semantic data without converting it to tabular data. This will be possible if we were to use the SANSA platform which supports the RDF data format. The SANSA framework (Lehmann et al., 2017) is a new platform that is built on the top of distributed systems such as Apache Spark. It enables us to make SPARQL queries and use data mining directly on semantic data in parallel mode. Unfortunately, this platform is still new and we could not find sufficient references to run it locally.

More broadly, research is also needed to apply negative or rare ARM (Kong et al., 2016; Luna, Romero, Romero, & Ventura, 2015) on semantic IoT datasets to detect anomalous events such as attacks or malicious datasets. Moreover, a rare ARM has been used to detect adverse drug reactions in the hospital system (Ji et al., 2013). Additionally, we will compare ARM as a classifier with classification algorithms such as decision tree or deep learning on IoT datasets. Further investigation and experimentation in using ARM as a feature selection step before using other machine learning algorithms is strongly recommended.

References

Abedjan, Z., & Naumann, F. (2011). *Context and target configurations for mining RDF data.* Paper presented at the Proceedings of the 1st international workshop on Search and mining entity-relationship data.

Abedjan, Z., & Naumann, F. (2013). Improving rdf data through association rule mining. *Datenbank-Spektrum, 13*(2), 111-120.

Afzali, M., Singh, N., & Kumar, S. (2016). *Hadoop-MapReduce: A platform for mining large datasets.* Paper presented at the Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on.

Atzmueller, M., Hayat, N., Trojahn, M., & Kroll, D. (2018). *Explicative human activity recognition using adaptive association rule-based classification.* Paper presented at the Future IoT Technologies (Future IoT), 2018 IEEE International Conference on.

Barati, M., Bai, Q., & Liu, Q. (2017). Mining semantic association rules from RDF data. *Knowledge-Based Systems, 133*, 183-196.

Barnaghi, P., Wang, W., Henson, C., & Taylor, K. (2012). Semantics for the Internet of Things: early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS), 8*(1), 1-21.

Calbimonte, J.-P., Jeung, H. Y., Corcho, O., & Aberer, K. (2012). Enabling query technologies for the semantic sensor web. *International Journal on Semantic Web and Information Systems, 8*(EPFL-ARTICLE-183971), 43-63.

Chen, X., & Zhou, L. (2015). *Design and implementation of an intelligent system for tourist routes recommendation based on Hadoop.* Paper presented at the Software Engineering and Service Science (ICSESS), 2015 6th IEEE International Conference on.

Cheng, B., Longo, S., Cirillo, F., Bauer, M., & Kovacs, E. (2015). *Building a big data platform for smart cities: Experience and lessons from santander.* Paper presented at the Big Data (BigData Congress), 2015 IEEE International Congress on.

de Medeiros, L. F., Priyatna, F., & Corcho, O. (2015). *MIRROR: Automatic R2RML mapping generation from relational databases.* Paper presented at the International Conference on Web Engineering.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51*(1), 107-113.

Gole, S., & Tidke, B. (2015). *Frequent Itemset Mining for Big Data in social media using ClustBigFIM algorithm.* Paper presented at the Pervasive Computing (ICPC), 2015 International Conference on.

Jafrin, R. (2015). *Data Annotation and Ontology Provisioning for Semantic Applications in Virtualized Wireless Sensor Networks.* Concordia University.

Kang, K. J., Ka, B., & Kim, S. J. (2012). A service scenario generation scheme based on association rule mining for elderly surveillance system in a smart home environment. *Engineering Applications of Artificial Intelligence, 25*(7), 1355-1364.

Khan, Z., Anjum, A., Soomro, K., & Tahir, M. A. (2015). Towards cloud based big data analytics for smart future cities. *Journal of Cloud Computing, 4*(1), 2.

Kong, H., Jong, C., & Ryang, U. (2016). Rare Association Rule Mining for Network Intrusion Detection. *arXiv preprint arXiv:1610.04306*.

Korzun, D. G., Balandin, S. I., & Gurtov, A. V. (2013). Deployment of Smart Spaces in Internet of Things: Overview of the design challenges *Internet of Things, Smart Spaces, and Next Generation Networking* (pp. 48-59): Springer.

Kothari, J. V., & Patel, K. (2015). Probability-based Incremental Association Rules Algorithm Using Hashing Technique. *International Journal of Advance Research in Computer Science and Management Studies, 3*(2), 321-327.

Larose, D. T., & Larose, C. D. (2014). *Discovering knowledge in data: an introduction to data mining*: John Wiley & Sons.

Li, H., Wang, Y., Zhang, D., Zhang, M., & Chang, E. Y. (2008). *Pfp: parallel fp-growth for query recommendation.* Paper presented at the Proceedings of the 2008 ACM conference on Recommender systems.

Li, S., & Zhou, C. (2017). *Research on power demand response of smart home control system based on association rules algorithm.* Paper presented at the Computer and Communications (ICCC), 2017 3rd IEEE International Conference on.

Liang, Y.-h., & Wu, S.-y. (2015). *Sequence-growth: A scalable and effective frequent itemset mining algorithm for big data based on MapReduce framework.* Paper presented at the Big Data (BigData Congress), 2015 IEEE International Congress on.

Lilleberg, I. (2015). Logging Web Behaviour for Association Rule Mining.

Lin, M.-Y., Lee, P.-Y., & Hsueh, S.-C. (2012). *Apriori-based frequent itemset mining algorithms on MapReduce.* Paper presented at the Proceedings of the 6th international conference on ubiquitous information management and communication.

Lin, X. (2014). *Mr-apriori: Association rules algorithm based on mapreduce.* Paper presented at the Software Engineering and Service Science (ICSESS), 2014 5th IEEE International Conference on.

Ma, B. L. W. H. Y., & Liu, B. (1998). *Integrating classification and association rule mining.* Paper presented at the Proceedings of the fourth international conference on knowledge discovery and data mining.

Maarala, A. I., Su, X., & Riekki, J. (2014). *Semantic data provisioning and reasoning for the internet of things.* Paper presented at the Internet of Things (IOT), 2014 International Conference on the.

Makanju, A., Farzanyar, Z., An, A., Cercone, N., Hu, Z. Z., & Hu, Y. (2016). *Deep parallelization of parallel FP-growth using parent-child MapReduce.* Paper presented at the Big Data (Big Data), 2016 IEEE International Conference on.

Mezghani, E., Exposito, E., Drira, K., Da Silveira, M., & Pruski, C. (2015). A semantic big data platform for integrating heterogeneous wearable data in healthcare. *Journal of medical systems, 39*(12), 185.

Naacke, H., Curé, O., & Amann, B. (2016). SPARQL query processing with Apache Spark. *arXiv preprint arXiv:1604.08903*.

Nguyen, D., Nguyen, L. T., Vo, B., & Pedrycz, W. (2016). Efficient mining of class association rules with the itemset constraint. *Knowledge-Based Systems, 103*, 73-88.

Nithya, N., & Duraiswamy, K. (2014). Gain ratio based fuzzy weighted association rule mining classifier for medical diagnostic interface. *Sadhana, 39*(1), 39-52.

Onal, A. C., Sezer, O. B., Ozbayoglu, M., & Dogdu, E. (2017). *Weather data analysis and sensor fault detection using an extended IoT framework with semantics, big data, and machine learning.* Paper presented at the Big Data (Big Data), 2017 IEEE International Conference on.

Park, J. S., Chen, M.-S., & Yu, P. S. (1995). *An effective hash-based algorithm for mining association rules* (Vol. 24): ACM.

Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*: Elsevier.

Prasad, H. M. (2017). *Revamped Market-Basket Analysis using In-Memory Computation framework.* Paper presented at the Intelligent Systems and Control (ISCO), 2017 11th International Conference on.

Qiu, H., Gu, R., Yuan, C., & Huang, Y. (2014). *Yafim: a parallel frequent itemset mining algorithm with spark.* Paper presented at the Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International.

Ramezani, R., Saraee, M., & Nematbakhsh, M. (2014). SWApriori: a new approach to mining Association Rules from Semantic Web Data. *Journal of Computing and Security*.

Rathore, M. M., Ahmad, A., Paul, A., & Rho, S. (2016). *Urban Planning and Building Smart Cities based on the Internet of Things using Big Data Analytics* (Vol. 101).

Ristoski, P., & Paulheim, H. (2016). Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Web semantics: science, services and agents on the World Wide Web, 36*, 1-22.

Rozik, A., Tolba, A., & El-Dosuky, M. (2016). Design and implementation of the sense egypt platform for real-time analysis of iot data streams. *Advances in Internet of Things, 6*(04), 65.

Schätzle, A., Przyjaciel-Zablocki, M., Skilevic, S., & Lausen, G. (2016). S2RDF: RDF querying with SPARQL on spark. *Proceedings of the VLDB Endowment, 9*(10), 804-815.

Segaran, T., Evans, C., & Taylor, J. (2009). *Programming the Semantic Web: Build Flexible Applications with Graph Data*: " O'Reilly Media, Inc.".

Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J.-P., Riahi, M., . . . Žarko, I. P. (2015). Openiot: Open source internet-of-things in the cloud *Interoperability and open-source solutions for the internet of things* (pp. 13-25): Springer.

Tsai, C.-W., Lai, C.-F., Chiang, M.-C., & Yang, L. T. (2014). Data mining for Internet of Things: A survey. *IEEE Communications Surveys and Tutorials, 16*(1), 77-97.

Tsay, L.-S., Sukumar, S. R., & Roberts, L. W. (2015). *Scalable association rule mining with predication on semantic representations of data.* Paper presented at the Technologies and Applications of Artificial Intelligence (TAAI), 2015 Conference on.

Yassine, A., Singh, S., & Alamri, A. (2017). Mining human activity patterns from smart home big data for health care applications. *IEEE Access, 5*, 13131-13141.

Yoshimura, Y., Sobolevsky, S., Bautista Hobin, J. N., Ratti, C., & Blat, J. (2016). Urban association rules: uncovering linked trips for shopping behavior. *Environment and Planning B: Planning and Design*, 0265813516676487.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., . . . Stoica, I. (2012). *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing.* Paper presented at the Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation.

Zaïane, O. R., El-Hajj, M., & Lu, P. (2001). *Fast parallel association rule mining without candidacy generation.* Paper presented at the Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on.

Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering, 12*(3), 372-390.

Zaki, M. J., & Gouda, K. (2003). *Fast vertical mining using diffsets.* Paper presented at the Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining.

Zecevic, P., & Bonaci, M. (2016). *Spark in Action*: Manning Publications Co.

Zhou, L., & Wang, X. (2014). Research of the FP-Growth Algorithm Based on Cloud Environments. *JSW, 9*(3), 676-683.

Zhou, L., Zhong, Z., Chang, J., Li, J., Huang, J. Z., & Feng, S. (2010). *Balanced parallel fp-growth with mapreduce.* Paper presented at the Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on.