Algortihms for Massive Datasets Report

Scalable Market Basket Analysis Using PySpark On IMDB Dataset

Data Science and Economics

Nicholas Carp

Department of Economics and Quantitative Methods
Università degli Studi di Milano

25-06-2021

Contents

1. Introduction

Market Basket analysis has become very important tool in retail and e-commerce, for analysing relationships between items. In particular is used to analyze customer buying habits by finding associations between the different items that customers place in their shopping baskets. The discovery of these associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers.

In this case we use a dataset composed by movies, series, actors, directors, ratings etc. Our goal is to see which actors are most frequently used in different baskets (movies). To achieve our goal we use Frequent Pattern algorithm.

2. Dataset and Preprocessing

2.1 Dataset

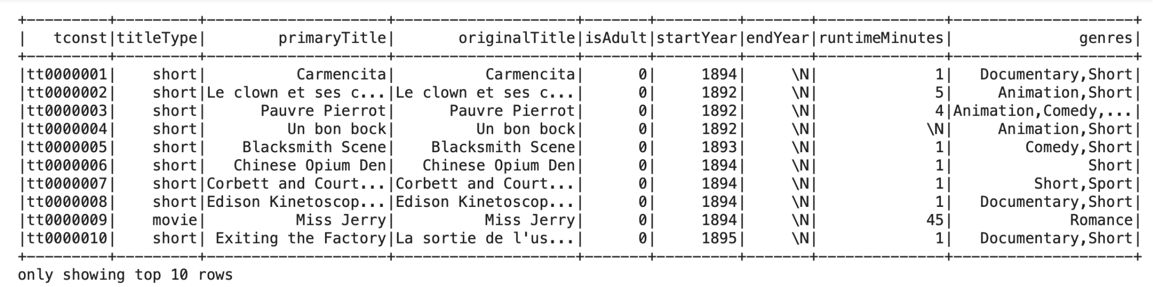
The data set that we analysed is IMDB data set and you can find it on Kaggle at this LINK. The data set is quite big (1.4 Gb) but small compared to usual data sets in big data analysis.

Is composed by 4 tables like it follows:

Data Composition	
Title Basics	6+ Million rows x 9 columns
Title Principals	36+ Million rows x 6 columns
Title Ratings	996+ K rows x 3 columns
Name Basics	9+ Million rows x 6 columns

We can display the data sets trough the following example tables: In the Title Basics table we are interested in the attribute like: the movie identifier(tconst), title, year, duration and genre.

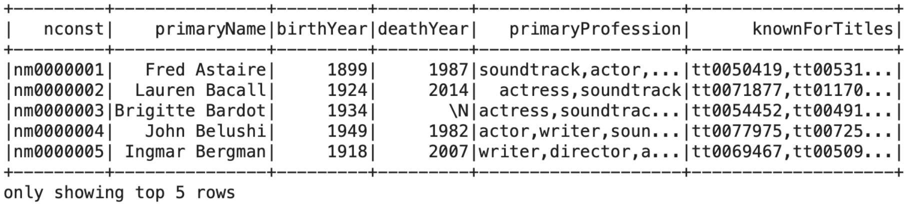
TITLE BASICS



```
+---------+---------+--------------------+--------------------+-------+---------+-------+--------------+--------------------+
|   tconst|titleType|        primaryTitle|       originalTitle|isAdult|startYear|endYear|runtimeMinutes|              genres|
+---------+---------+--------------------+--------------------+-------+---------+-------+--------------+--------------------+
|tt0000001|    short|          Carmencita|          Carmencita|      0|     1894|     \N|             1|   Documentary,Short|
|tt0000002|    short|Le clown et ses c...|Le clown et ses c...|      0|     1892|     \N|             5|     Animation,Short|
|tt0000003|    short|      Pauvre Pierrot|      Pauvre Pierrot|      0|     1892|     \N|             4|Animation,Comedy,...|
|tt0000004|    short|         Un bon bock|         Un bon bock|      0|     1892|     \N|            \N|     Animation,Short|
|tt0000005|    short|    Blacksmith Scene|    Blacksmith Scene|      0|     1893|     \N|             1|        Comedy,Short|
|tt0000006|    short|   Chinese Opium Den|   Chinese Opium Den|      0|     1894|     \N|             1|               Short|
|tt0000007|    short|Corbett and Court...|Corbett and Court...|      0|     1894|     \N|             1|         Short,Sport|
|tt0000008|    short|Edison Kinetoscop...|Edison Kinetoscop...|      0|     1894|     \N|             1|   Documentary,Short|
|tt0000009|    movie|          Miss Jerry|          Miss Jerry|      0|     1894|     \N|            45|             Romance|
|tt0000010|    short| Exiting the Factory|La sortie de l'us...|      0|     1895|     \N|             1|   Documentary,Short|
+---------+---------+--------------------+--------------------+-------+---------+-------+--------------+--------------------+
only showing top 10 rows
```

Name Basics is the table representing the persons, we are interested in their name identifiers(nconst), their names and their date of birth(and eventually death).

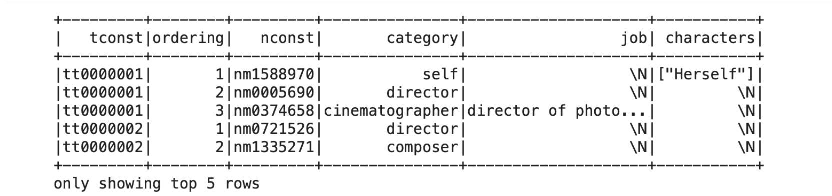
NAME BASICS



```
+---------+---------------+---------+---------+--------------------+--------------------+
|   nconst|    primaryName|birthYear|deathYear|   primaryProfession|      knownForTitles|
+---------+---------------+---------+---------+--------------------+--------------------+
|nm0000001|   Fred Astaire|     1899|     1987|soundtrack,actor,...|tt0050419,tt00531...|
|nm0000002|  Lauren Bacall|     1924|     2014|  actress,soundtrack|tt0071877,tt01170...|
|nm0000003|Brigitte Bardot|     1934|       \N|actress,soundtrac...|tt0054452,tt00491...|
|nm0000004|   John Belushi|     1949|     1982|actor,writer,soun...|tt0077975,tt00725...|
|nm0000005| Ingmar Bergman|     1918|     2007|writer,director,a...|tt0069467,tt00509...|
+---------+---------------+---------+---------+--------------------+--------------------+
only showing top 5 rows
```

In Title principals we can find the matching between the movie and the role of a person in that film (actor,director, screenwriter etc.) trough the name identifier. Other attributes (jobs and charachters) are not important for our analysis.

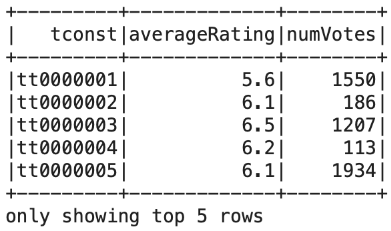
TITLE PRINCIPALS



```
+---------+--------+---------+---------------+--------------------+-----------+
|   tconst|ordering|   nconst|       category|                 job| characters|
+---------+--------+---------+---------------+--------------------+-----------+
|tt0000001|       1|nm1588970|           self|                  \N|["Herself"]|
|tt0000001|       2|nm0005690|       director|                  \N|         \N|
|tt0000001|       3|nm0374658|cinematographer|director of photo...|         \N|
|tt0000002|       1|nm0721526|       director|                  \N|         \N|
|tt0000002|       2|nm1335271|       composer|                  \N|         \N|
+---------+--------+---------+---------------+--------------------+-----------+
only showing top 5 rows
```

In the ratings table we find the film identifier, the average rating score and the number of voters.

TITLE RATINGS



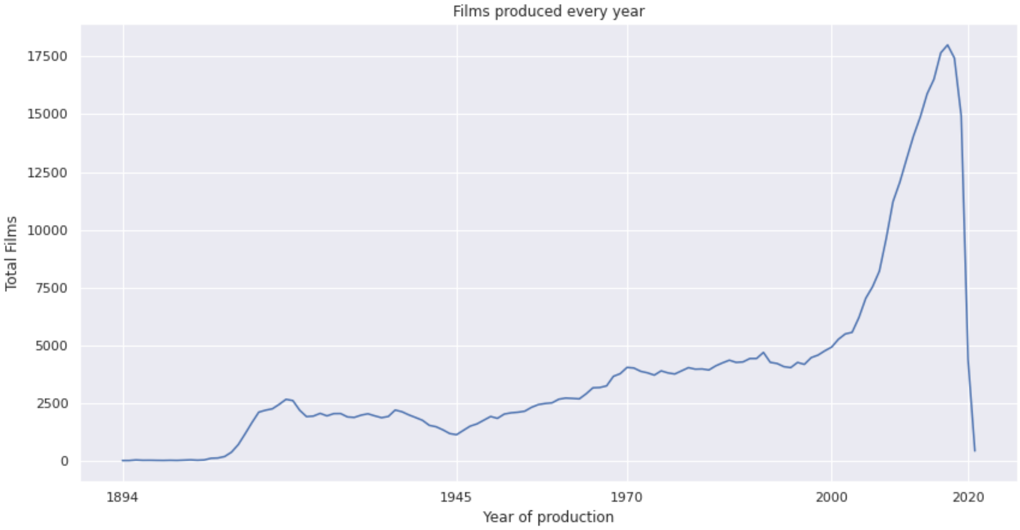
```
+---------+-------------+--------+
|   tconst|averageRating|numVotes|
+---------+-------------+--------+
|tt0000001|          5.6|    1550|
|tt0000002|          6.1|     186|
|tt0000003|          6.5|    1207|
|tt0000004|          6.2|     113|
|tt0000005|          6.1|    1934|
+---------+-------------+--------+
only showing top 5 rows
```

2.2 Preprocessing and EDA

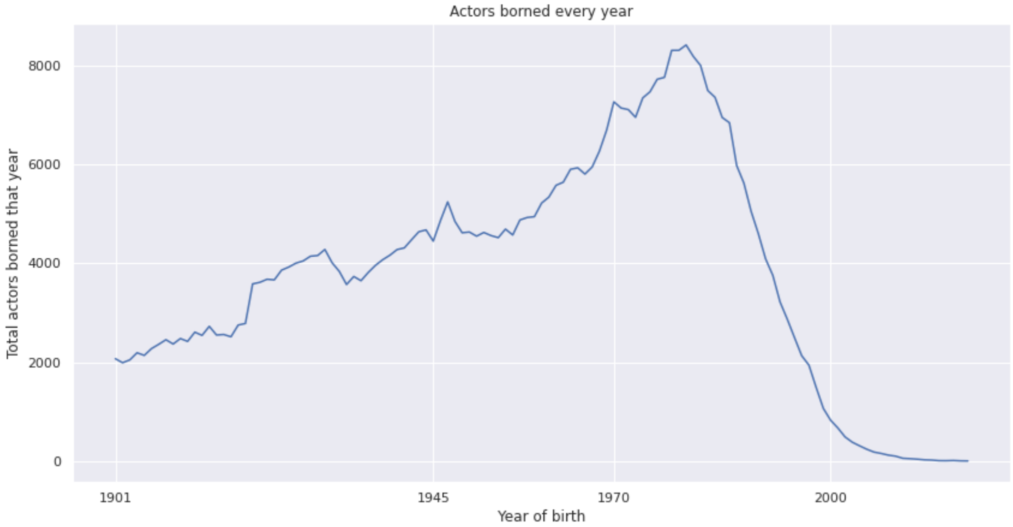
We start our preprocessing phase transforming our tsv files into Spark Dataframes. Then we explore our data with some queries, for example retrieving the movies with highest average score and a consistent number of reviews (>200) because it would be more significant.

Title	Votes	Rating
Randhawa	816	9.8
Gini Helida Kathe	425	9.8
Square One	498	9.8
Hare Krishna!	1200	9.7
Fan	1010	9.6
Android Kunjappan Version 5.25	1175	9.6
Mama's Heart. Gongadze	502	9.6
Retrocausality	1420	9.5
Safe	1017	9.5
Yeh Suhaagraat Impossible	635	9.5
The Brighton Miracle	617	9.5
Svet Koji Nestaje	245	9.5
Jibon Theke Neya	1651	94

Or we can display the number of movies produced every year and notice the disruptive effect that Covid-19 had on this industry in 2020/2021, precisely we have 4389 for 2020 (back at 1970 levels), and 414 for 2021.



And ultimately we can see the number of actors born every year and notice for example that it has quite the same pattern of the industry expansion, indicating obviously during time industry became bigger and needed more actors. We can also see that it has a peak around 1980, so the age that is playing an important role in this industry is 41 years old. Then for young actors (<30 years old) of course there is less presence in movies because their total number of performances are not all already happened.



3. Algorithms and Implementation

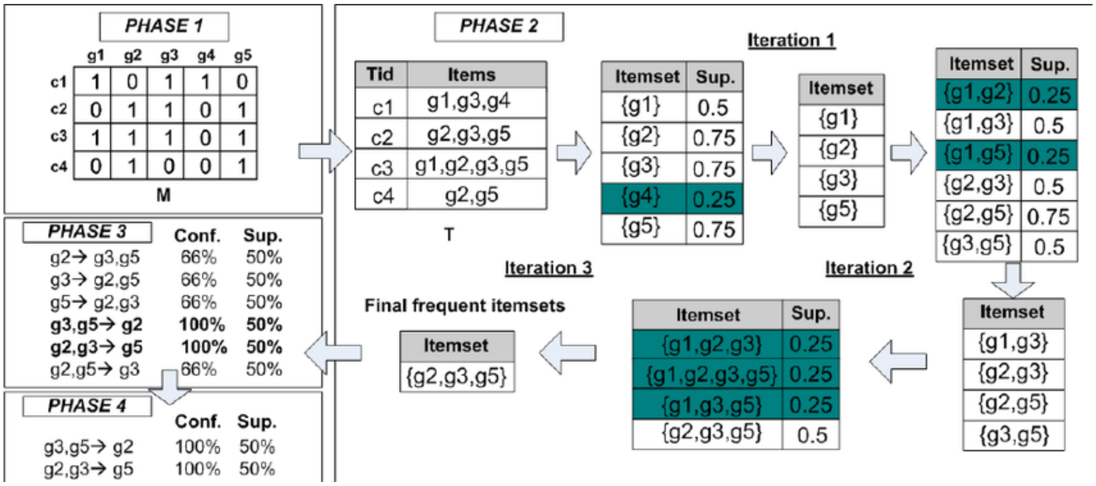
Market basket analysis is the analysis of any collection of items to identify items frequencies and association rules. The necessary for MBA are some transactions containing groups of items.

The advantage of association analysis is that it uses unsupervised learning to learn hidden patterns so there is limited need for data preparation and feature engineering and relatively easy to explain to non-technical people. It is a good start for certain cases of data exploration and can point the way for a deeper dive into the data using other approaches.

3.1 Apriori and FPGrowth Algorithms

To compute the association rules in MBA Apriori algorithm can be used. Apriori algorithm has some disadvantages such as the fact of being computationally expensive or taking long time to finish a run. It happens because the algorithm needs to iterate many times over the whole dataset to generate the itemsets. Here it is an example of how Apriori algorithm works.

EXAMPLE OF APRIORI ALGORITHM



	g1	g2	g3	g4	g5
c1	1	0	1	1	0
c2	0	1	1	0	1
c3	1	1	1	0	1
c4	0	1	0	0	1

Tid	Items
c1	g1,g3,g4
c2	g2,g3,g5
c3	g1,g2,g3,g5
c4	g2,g5

Itemset	Sup.
{g1}	0.5
{g2}	0.75
{g3}	0.75
{g4}	0.25
{g5}	0.75

Itemset
{g1}
{g2}
{g3}
{g5}

Itemset	Sup.
{g1,g2}	0.25
{g1,g3}	0.5
{g1,g5}	0.25
{g2,g3}	0.5
{g2,g5}	0.75
{g3,g5}	0.5

Itemset
{g1,g3}
{g2,g3}
{g2,g5}
{g3,g5}

Itemset	Sup.
{g1,g2,g3}	0.25
{g1,g2,g3,g5}	0.25
{g1,g3,g5}	0.25
{g2,g3,g5}	0.5

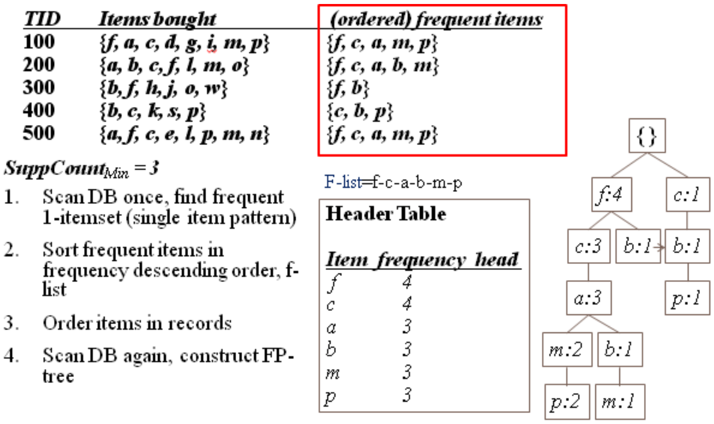
Itemset
{g2,g3,g5}

	Conf.	Sup.
g2→ g3,g5	66%	50%
g3→ g2,g5	66%	50%
g5→ g2,g3	66%	50%
g3,g5→ g2	**100%**	**50%**
g2,g3→ g5	**100%**	**50%**
g2,g5→ g3	66%	50%

	Conf.	Sup.
g3,g5→ g2	100%	50%
g2,g3→ g5	100%	50%

To overcome the disadvantages of the Apriori algorithm, it was created an improvement of the algorithm called FP(Frequent Pattern)-Growth. Instead of iterating over the whole data set like the Apriori algorithm, FP-Growth iterates only twice and uses a tree structure (FP-Tree) to storage the information. Each tree node represents an item and the number of baskets that are originated by that node. Each nodes connection represents an itemset.

Firstly, the algorithm will be iterating over the dataset and will count the absolute frequency of each item. The items will be sort by frequency (support count).



TID	*Items bought*	*(ordered) frequent items*
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o, w*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}

SuppCount$_{Min}$ = 3

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Order items in records
4. Scan DB again, construct FP-tree

F-list=f-c-a-b-m-p

Header Table

Item	*frequency*	*head*
f	4	
c	4	
a	3	
b	3	
m	3	
p	3	

Next step is to build a table, considering support count in ascending order, and to do the frequent pattern generation for each item.

Last step is building a tree. Starting by the root, that is the empty basket, items will be added by descending order of support count and also considering the content of the baskets.

3.2 Implementation

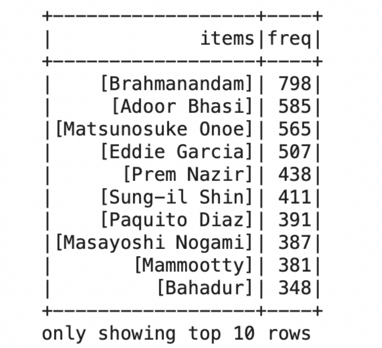
In this case we will use the movies as transactions, and the set of actors that play in that film as basket. In the initial phase support and confidence are set, confidence is minimum confidence of the rule to be included in our results. Confidence divides the number of transactions involving both items by the number of transactions involving only one item.

The implementation approach is to find most frequent singular items, then most frequent sets two items and then display a prediction table that shows predictions that have more confidence. In the end we will look for a specific actor as and see what is the basket most frequently associated with this actor. Ideally we can explore any actor association.

4. Experimental Results

4.1 Results and Considerations

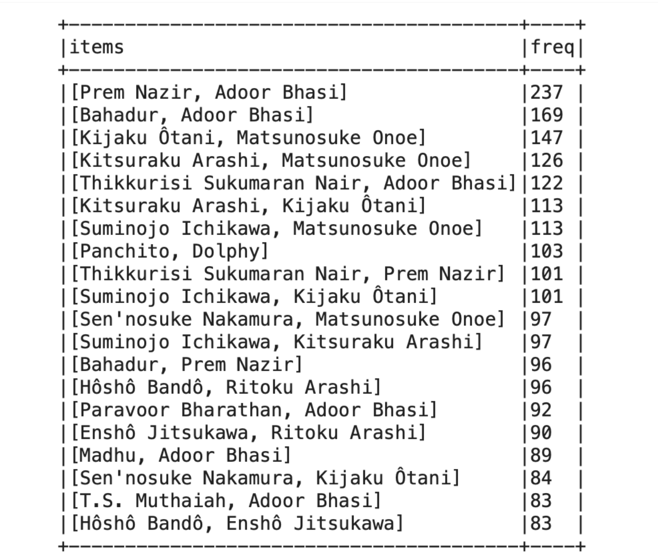
We built the table containing the identificative serial for the film and the basket of actors that played in that film. We apply FP-Growth with support 0.00001 and minimum confidence of 0.1. First output is most frequent items table, and displays the actors with highest number of movies.



```
+------------------+----+
|             items|freq|
+------------------+----+
|    [Brahmanandam]| 798|
|     [Adoor Bhasi]| 585|
|[Matsunosuke Onoe]| 565|
|    [Eddie Garcia]| 507|
|      [Prem Nazir]| 438|
|    [Sung-il Shin]| 411|
|    [Paquito Diaz]| 391|
|[Masayoshi Nogami]| 387|
|       [Mammootty]| 381|
|         [Bahadur]| 348|
+------------------+----+
only showing top 10 rows
```

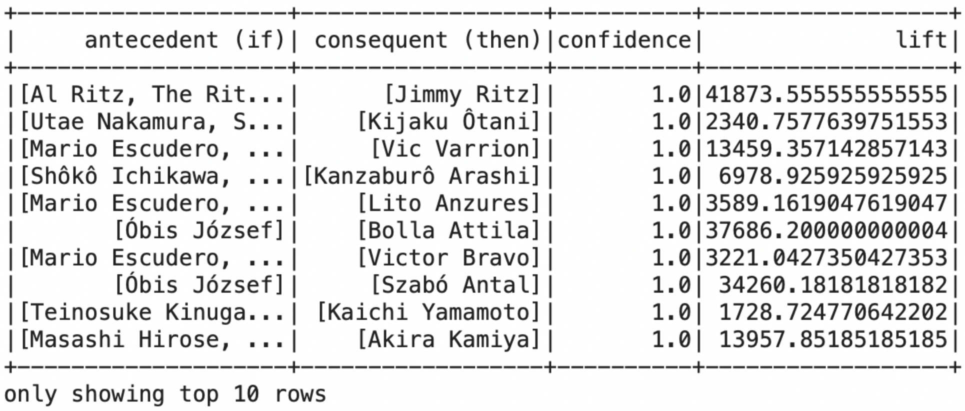
We can see that are mostly actors from Asia (India,Japan,Philippines) and some of them already died.

Then we also view the pairs of actors that appear more frequently.



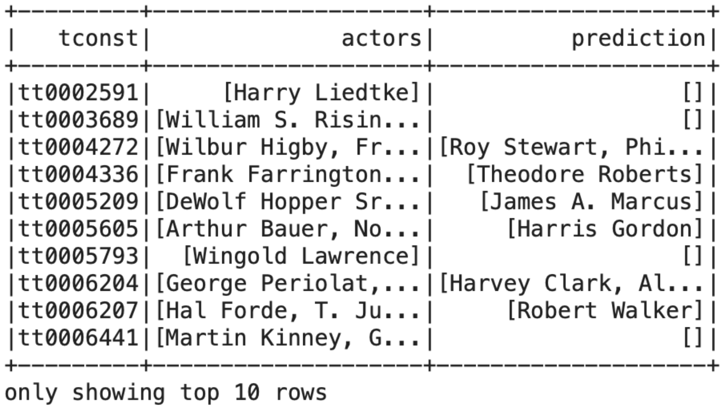
```
+----------------------------------------+----+
|items                                   |freq|
+----------------------------------------+----+
|[Prem Nazir, Adoor Bhasi]               |237 |
|[Bahadur, Adoor Bhasi]                  |169 |
|[Kijaku Ôtani, Matsunosuke Onoe]        |147 |
|[Kitsuraku Arashi, Matsunosuke Onoe]    |126 |
|[Thikkurisi Sukumaran Nair, Adoor Bhasi]|122 |
|[Kitsuraku Arashi, Kijaku Ôtani]        |113 |
|[Suminojo Ichikawa, Matsunosuke Onoe]   |113 |
|[Panchito, Dolphy]                      |103 |
|[Thikkurisi Sukumaran Nair, Prem Nazir] |101 |
|[Suminojo Ichikawa, Kijaku Ôtani]       |101 |
|[Sen'nosuke Nakamura, Matsunosuke Onoe] |97  |
|[Suminojo Ichikawa, Kitsuraku Arashi]   |97  |
|[Bahadur, Prem Nazir]                   |96  |
|[Hôshô Bandô, Ritoku Arashi]            |96  |
|[Paravoor Bharathan, Adoor Bhasi]       |92  |
|[Enshô Jitsukawa, Ritoku Arashi]        |90  |
|[Madhu, Adoor Bhasi]                    |89  |
|[Sen'nosuke Nakamura, Kijaku Ôtani]     |84  |
|[T.S. Muthaiah, Adoor Bhasi]            |83  |
|[Hôshô Bandô, Enshô Jitsukawa]          |83  |
+----------------------------------------+----+
```

In the association rules table we have the basket of antecedents, the consequent actor, the confidence level of the rule and the lift. We display an example sorted by confidence.



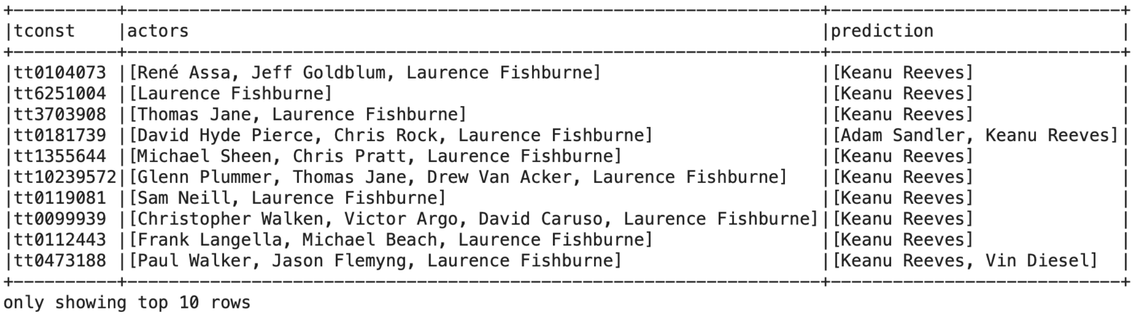
```
+--------------------+------------------+----------+------------------+
|     antecedent (if)| consequent (then)|confidence|              lift|
+--------------------+------------------+----------+------------------+
|[Al Ritz, The Rit...|      [Jimmy Ritz]|       1.0|41873.555555555555|
|[Utae Nakamura, S...|    [Kijaku Ôtani]|       1.0|2340.7577639751553|
|[Mario Escudero, ...|     [Vic Varrion]|       1.0|13459.357142857143|
|[Shôkô Ichikawa, ...|[Kanzaburô Arashi]|       1.0| 6978.925925925925|
|[Mario Escudero, ...|    [Lito Anzures]|       1.0|3589.1619047619047|
|       [Óbis József]|    [Bolla Attila]|       1.0|37686.200000000004|
|[Mario Escudero, ...|    [Victor Bravo]|       1.0|3221.0427350427353|
|       [Óbis József]|     [Szabó Antal]|       1.0| 34260.18181818182|
|[Teinosuke Kinuga...| [Kaichi Yamamoto]|       1.0| 1728.724770642202|
|[Masashi Hirose, ...|    [Akira Kamiya]|       1.0| 13957.85185185185|
+--------------------+------------------+----------+------------------+
only showing top 10 rows
```

We can also make predictions based on the association rules. Some Predictions are not present because the support or the confidence of the association rule was too low and so it is not computed.



```
+---------+--------------------+--------------------+
|   tconst|              actors|          prediction|
+---------+--------------------+--------------------+
|tt0002591|     [Harry Liedtke]|                  []|
|tt0003689|[William S. Risin...|                  []|
|tt0004272|[Wilbur Higby, Fr...|[Roy Stewart, Phi...|
|tt0004336|[Frank Farrington...|  [Theodore Roberts]|
|tt0005209|[DeWolf Hopper Sr...|   [James A. Marcus]|
|tt0005605|[Arthur Bauer, No...|     [Harris Gordon]|
|tt0005793|  [Wingold Lawrence]|                  []|
|tt0006204|[George Periolat,...|[Harvey Clark, Al...|
|tt0006207|[Hal Forde, T. Ju...|     [Robert Walker]|
|tt0006441|[Martin Kinney, G...|                  []|
+---------+--------------------+--------------------+
only showing top 10 rows
```

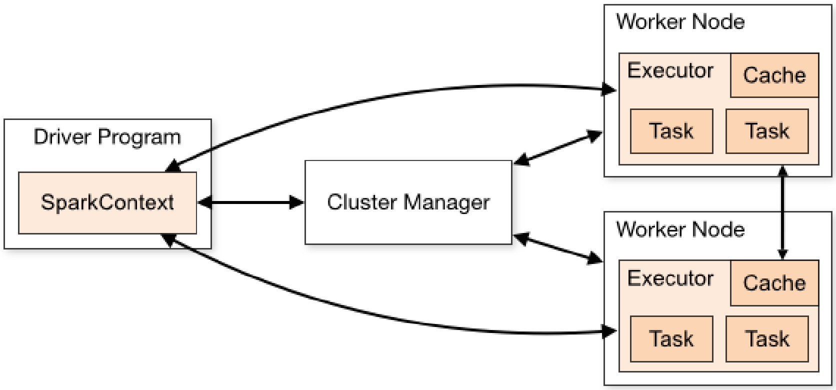
We selected a specific actor, in this case is "Keanu Reeves", and show which baskets of actors has Keanu as prediction.



```
+----------+-------------------------------------------------------------------+----------------------------+
|tconst    |actors                                                             |prediction                  |
+----------+-------------------------------------------------------------------+----------------------------+
|tt0104073 |[René Assa, Jeff Goldblum, Laurence Fishburne]                     |[Keanu Reeves]              |
|tt6251004 |[Laurence Fishburne]                                               |[Keanu Reeves]              |
|tt3703908 |[Thomas Jane, Laurence Fishburne]                                  |[Keanu Reeves]              |
|tt0181739 |[David Hyde Pierce, Chris Rock, Laurence Fishburne]                |[Adam Sandler, Keanu Reeves]|
|tt1355644 |[Michael Sheen, Chris Pratt, Laurence Fishburne]                   |[Keanu Reeves]              |
|tt10239572|[Glenn Plummer, Thomas Jane, Drew Van Acker, Laurence Fishburne]   |[Keanu Reeves]              |
|tt0119081 |[Sam Neill, Laurence Fishburne]                                    |[Keanu Reeves]              |
|tt0099939 |[Christopher Walken, Victor Argo, David Caruso, Laurence Fishburne]|[Keanu Reeves]              |
|tt0112443 |[Frank Langella, Michael Beach, Laurence Fishburne]                |[Keanu Reeves]              |
|tt0473188 |[Paul Walker, Jason Flemyng, Laurence Fishburne]                   |[Keanu Reeves, Vin Diesel]  |
+----------+-------------------------------------------------------------------+----------------------------+
only showing top 10 rows
```

4.2 Scalability

Our data set is not so big (1.44 Gb), but if we had used pandas dataframes for example, our notebook would have crashed. However our algorithm works even with a bigger data set thanks to Spark Cluster and Spark Dataframes.



Apache Spark follows a master/slave architecture, a Spark Application consists of a Driver Program and a group of Executors on the cluster.

The Driver is a process that executes the main program of your Spark application and is responsible for converting a user application to smaller execution units called tasks and then schedules them to run with a cluster manager on executors.

The Driver also creates the SparkContext, SparkContext is used by the Driver Process in order to establish a communication with the cluster and the resource managers in order to coordinate and execute jobs. Spark Driver contains various components (DAGScheduler, TaskScheduler, BackendScheduler and BlockManager) responsible for the translation of spark user code into spark jobs executed on the cluster.

The executors are processes running on the worker nodes of the cluster which are responsible for executing the tasks the driver process has assigned to them. The cluster manager (such as Mesos or YARN) is a process that controls, governs, and reserves computing resources in the form of containers on the cluster. These containers are reserved by request of Application Master and are allocated to Application Master when they are released or available.

Spark Dataframes are data organized into named columns, like a table in a relational database. DataFrames can be constructed from a wide array of sources such as: structured data files, distributed file systems, parquet files, tables in Hive, external databases, or existing RDDs. DataFrame in Spark allows developers to impose a structure onto a distributed collection of data, allowing higher-level abstraction.

5. Conclusions

Resuming what we did, we set a Spark application, built the data frames, constructed the baskets, applied FP-Growth algorithm to determine most frequent item and most frequent set. Analysed association rules, predictions and customized exploration of them. Then we made some considerations on scalability and the fact that we could use this method with any dataset thanks to Spark. Of course we could go deeper in exploration doing the same research on other role in the movie, or maybe doing some analysis on series and shorts.

6. Declaration

“I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.“