MARKET BASKET ANALYSIS
FINDING FREQUENT ITEMSETS IN THE IMDB DATASET

ABSTRACT

This article is supposed to demonstrate an approach of solving a classical class of problems called Market Basket analyses. The main idea behind this problem – is to find items that are frequently bought together in one consumer basket using efficient computational algorithms. The naïve approach of cycling over each item and each available itemset is a memory expensive task and could probably be not executed on a standard machine. In order to accomplish the computation using limited resources we demonstrate a MapReduce approach that allows us to find the most frequent pairs of actors appearing together on screen using IMDB dataset.

Keywords: Market basket analysis · MapReduce · Apriori Algorithm

Contents

1. Introduction

Market basket analyses is usually performed in order to understand what products customers usually buy together. We can relate to it as finding complementary products in a consumer basket.

This information could be very beneficial for several reasons:

- we can predict the rise or fall in demand for one product by the price change of another
- we can offer the best arrangement of goods in stores
- we can improve recommendation systems
- we can improve marketing offers

At the same time, market basket analysis is not limited only for sales purposes. It can be also used in another context as well. In this paper we want to find actors who frequently play together in movies according to the IMDB dataset. This task could also be attributed to market basket analyses.

Performing market basket analyses, we are typically confronted with a lot of different items and itemsets that should be analyzed for finding useful information. This abundance of data imposes a restriction on using naïve approaches like cycling over each item and each itemset to find frequent joint appearances.

Thankfully, there are developed frameworks and algorithms of specific data transformations and parallel processing that allow us to fit into the computer hard drive and perform the needed computation.

In this paper, we are going to apply one of these algorithms, the Apriori algorithm, to find the most frequent pairs and triples of actors appearing together on screen according to the information from the IMDB dataset.

The dataset is going to be downloaded from Kaggle website using its API and the implementation of the algorithm will be done with the combination of PySpark and efficient_apriori module in Python.

Kaggle connection

We need to install Kaggle module using pip to be able to connect to its datasets using API. Once this step is done, we need to import credentials file(«kaggle.json») to our working directory and set up the Kaggle environment with standard Python os module. Once it's done, we are ready to download and unzip the IMDB dataset.

Initialize PySpark session

We need to install PySpark module of the latest version to be able to use it's all up-to-date methods and make their imports.

We initialize spark session by configurations:

- executor memory – 32GB
- driver memory – 128GB
- driver MaxResultSize – 24GB

Datasets

Now we dive deeper into the mentioned IMDB dataset. This dataset is a collection of several datasets covering the topics, movies, persons, ratings, etc. For performing the analyses, we are interested in the following files:

- titile.principals (alias principals)
- title.basics (alias movies)
- title.names (alias persons)

Table 1: 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 photography	\N
tt0000002	1	nm0721526	director	\N	\N
tt0000002	2	nm1335271	composer	\N	\N

Table 2: Movies

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 chiens	Le clown et ses chiens	0	1892	\N	5	Animation,Short
tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animation,Comedy,Romance
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

Table 3: Principals

nconst	primaryName	birthYear	deathYear	primaryProfession	knownForTitles
nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0050419,tt0053137,tt0043044,tt0072308
nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0117057,tt0037382,tt0071877,tt0038355
nm0000003	Brigitte Bardot	1934	\N	actress,soundtrack,producer	tt0049189,tt0059956,tt0054452,tt0057345
nm0000004	John Belushi	1949	1982	actor,writer,soundtrack	tt0078723,tt0080455,tt0072562,tt0077975
nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0050986,tt0083922,tt0069467,tt0050976

We can see there is lots of information about movies and principals involved in them: what type of a project it is, what is the name of the project, when it was filmed, who and what role they participated and so on. The main attributes that are useful for the purpose of finding frequent actors' pairs and triples are:

- movies.primaryTitle (the name of a movie as it's unique identifier)
- persons.primaryName (the name of an actor as it's unique identifier)
- principals.category (for filtering actors out of other professions)
- movies.titleType (for filtering movies out of other projects shown on screen)

Data exploration

As we have decided what datasets to use, what their columns to analyse, we have to make some exploration of our generated data to make sure it is suitable for applying Apriori algorithm:

- Analyses of None's: if data contains any missings
- Analyses of data types: if data contains incorrect data types

Table 4:Number of missings in data

column_name	n_missings
movies_titleType	0
movies_primaryTitle	0
persons_primaryName	0
principals.category	0

Table 5: Data types

column_name	column_type
movies_titleType	string
movies_primaryTitle	string
persons_primaryName	string
principals.category	string

We can conclude that there're no missings in the data and all the data types are correct, so there is no need for additional preprocessing.

2. Finding insights

As we our sure our dataset is generated, clean and is ready to act as an input for Apriori algorithm, we can count descriptive statistics of it and try to figure out some interesting insights.

- Our datasets consist of 1693314 rows
- There are 346947 unique actors(items) in the dataset
- There are 560852 unique movies(baskets) in the dataset

Top 10 actors with the biggest number of movies

Let's see top-10 actors with the biggest number of movies appearances and plot them on a histogram and as a wordcloud. This can be interpreted as the most frequent items(singletons) in all of the baskets.

Figure 1: Most frequent items

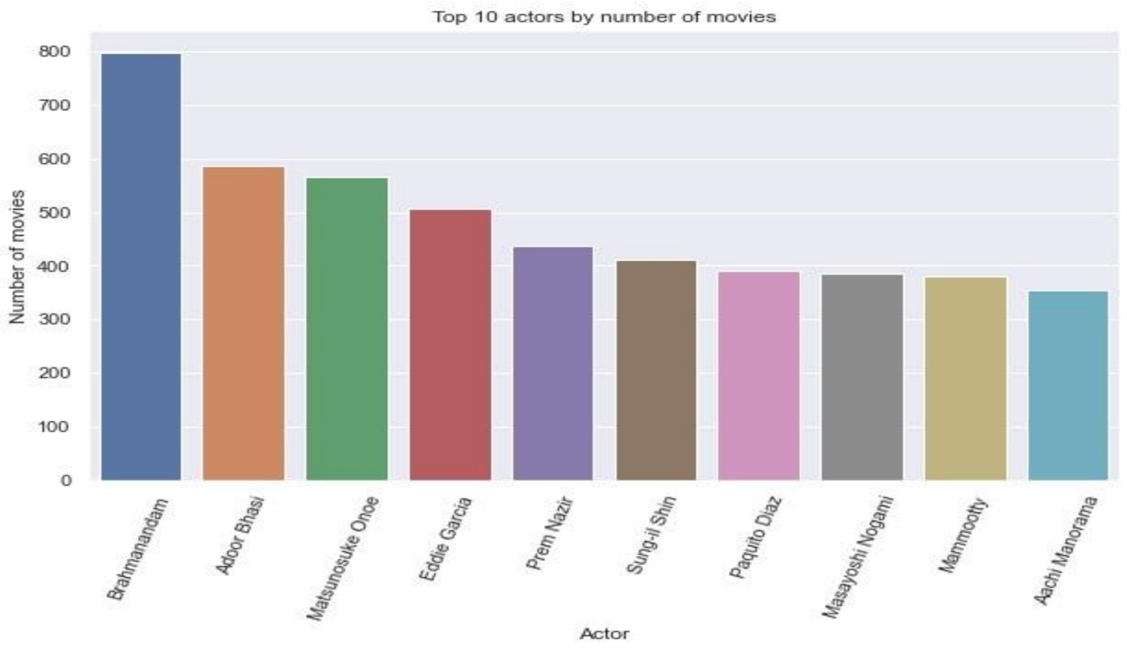
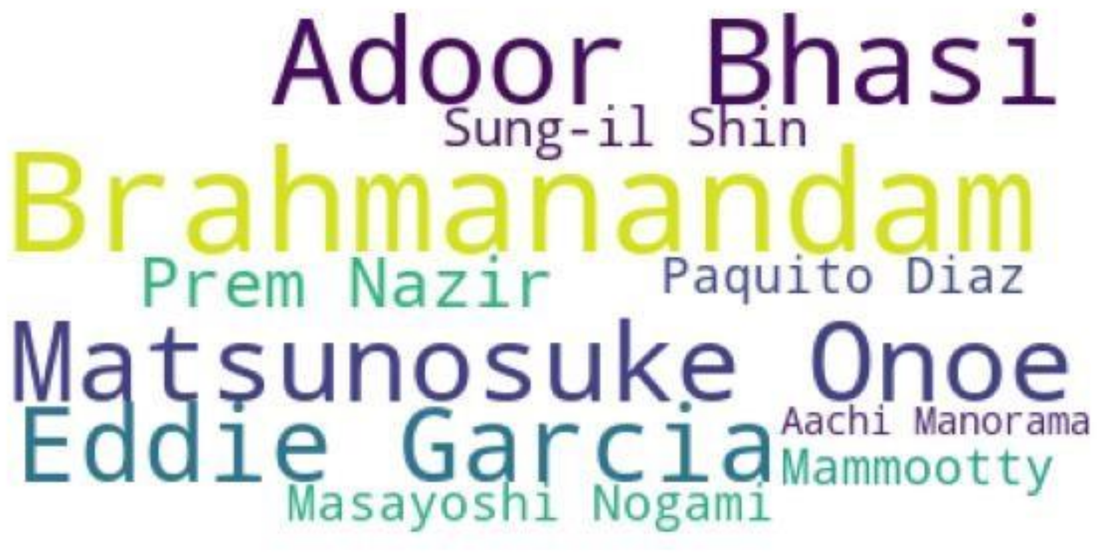


Figure 2: Most frequent items as a wordcloud

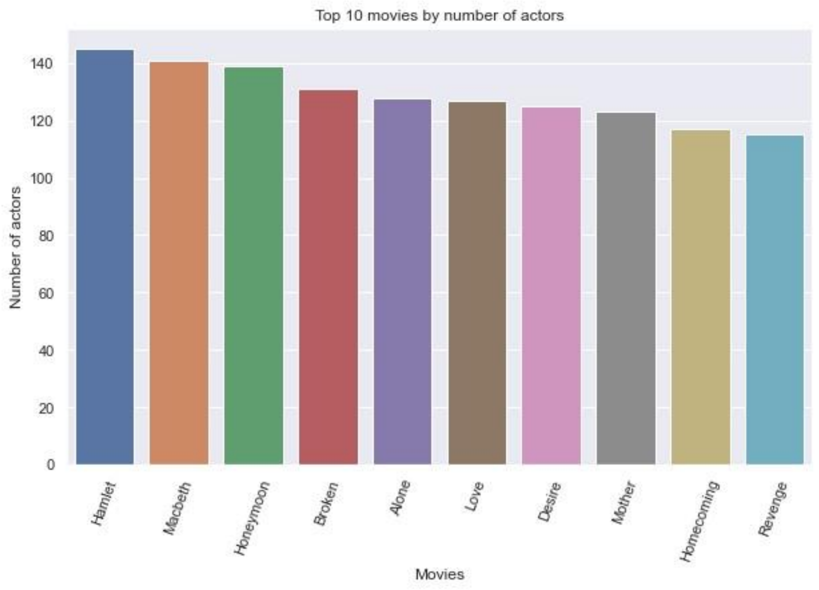


Actors who provided the biggest number of appearances are: Brahmanandam, Adoor Bhasi, Matsunosuke Onoe, Eddie Garcia, Prem Nazir, etc. Probably, some of these actors could have joint appearances and they may make pairs of most frequent itemsets.

Top 10 movies with the biggest number of actors

Let's see top-10 movies with the biggest number of actors and plot them on a histogram. This can be interpreted as baskets with the biggest number of items.

Figure 3: Baskets with the biggest number of items



As we see from the plot, the movie with the biggest number of actors is Hamlet: 145 different people have participated in this project.

3. Apriori Algorithm

On this step we have a clean dataset that shows the accordance between an actor and the movie he has played to. We need to use this information to find the most frequent pairs and triples of actors playing together in one movie.

As mentioned earlier, the basic solution would be to loop through all the movies(baskets) and inside the movies loop through all the combinations of actors and count them. The problem behind it is that this way is going to take much time or probably not be computed as the data will not fit the hard drive.

One of the solutions for the problem - Apriori algorithm based on MapReduce framework.

The idea of MapReduce framework is that we split the input data on some number of chunks, apply a map function that counts the number of unique items in each chunk, shuffle the data in from the chunks so that same unique items are grouped together, count the frequency of them in the whole dataset and return a {key:value} pair {unique item : its frequency in the dataset} as the output.

The idea of Apriori algorithm is that we do several MapReduce phases to calculate the most frequent itemsets. The number of phases is equal to the wanted length of a set. For example, if want to find most frequent pair of items – we apply two MapReduce phases, and if we want most frequent triple of items – we apply three MapReduce phases and so on.
The trick of the algorithm is that we ignore all sets that contain non-frequent pairs found on the current step while doing next step calculations. The number of appearances of each set in all the data is usually called «support» and the threshold for non-frequent pairs is called «min_support».

Once finding of frequent itemsets is done, we need can make association rules for the items that are written in a format: Actor X => Actor Y. It means you obtain a rule that tells you if an actor X has appeared in a movie, it is also likely for the actor Y to appear in a movie.

The measures to understand the power of association rules are confidence and lift.

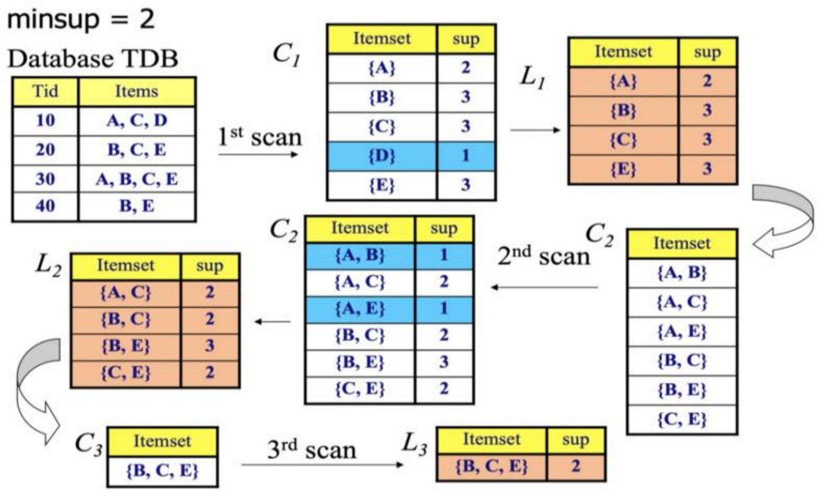
- Confidence tells a percentage of cases in which rule is valid. 100% confidence means association always occurs.
- Lift shows if items are independent or not. If lift of a rule is 1, then items are independent. Otherwise, if lift is high above 1, then items are strongly dependent.

Equation 1: lift formula

$$lift = \frac{\mathrm{P}(\mathrm{X}\ and\ Y)}{\mathrm{P}(\mathrm{X}) * P(Y)}$$

The illustration of Apriori algorithm is provided on the plot below.

Figure 4: Aprirori algorithm



Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

Itemset
{B, C, E}

Itemset	sup
{B, C, E}	2

Data organization

Data should be organized in such a way that you have a set of actors(items) on each line. Each of those sets contain actors(items) that appeared in the same movie.

1. We need to transform the dataframe from a {movie : actor} per line presentation to a {movie : list of actors played in this movie} presentation.
2. We need to transform the dataframe column with lists of actors played in the same movie into a list of lists of actors played in the same movie.
3. Efficient_apriori python module that makes the implementation of apriori algorithm takes a list of tuples as input, not a list of lists. We need to make this data transformation to provide correct input.

Algorithm implementation

Apriori function of efficient_apriori module that implements the algorithm has three hyperparameters:

- input data – our data organized on previous steps
- min_support – the threshold that shows minimum allowed frequency of an itemset
- min_confidence – allows to delete rules with low confidence

We assume that we want to set the threshold for support on a level of 100 and minimal confidence on a level of 100 to obtain all the rules.

Apriori function returns two variables: itemsets and rules.

- itemsets – desired frequent itemsets with different lengths of a set
- rules – calculated confidence and lift of association rules.

4. Conclusion

We have found most frequent pairs and triples of actors that appeared together in same movies.

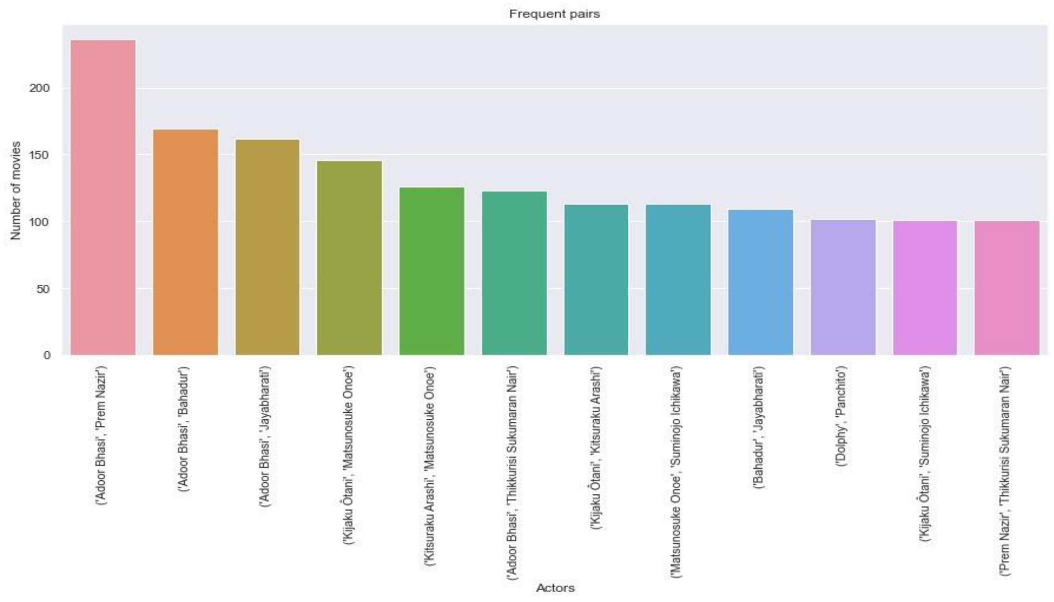
Frequent pairs

The most frequent pair is a pair of Indian actors Adoor Bhasi and Prem Nazir with 236 joint appearances. Adoor Bhasi was second in top-10 actors with the biggest number of movies and not surprisingly he became a part of this list. Overall, there are 12 pairs of actors that are frequent and are above the threshold. Lift is far above 1 for all of the rules so we can say the result is stable.

Table 6: Frequent pairs of actors

actors	joint_appearences	conf_1	conf_2	lift_1	lift_2
('Adoor Bhasi', 'Prem Nazir')	236	0,54	0,4	321,02	321,02
('Adoor Bhasi', 'Bahadur')	169	0,49	0,29	288,01	288,01
('Adoor Bhasi', 'Jayabharati')	162	0,53	0,28	317,09	317,09
('Kijaku Ôtani', 'Matsunosuke Onoe')	146	0,29	0,91	623,21	623,21
('Kitsuraku Arashi', 'Matsunosuke Onoe')	126	0,25	0,98	672,3	672,3
('Adoor Bhasi', 'Thikkurisi Sukumaran Nair')	123	0,47	0,21	280,57	280,57
('Kijaku Ôtani', 'Kitsuraku Arashi')	113	0,88	0,71	1914,31	1914,31
('Matsunosuke Onoe', 'Suminojo Ichikawa')	113	0,97	0,22	659,62	659,62
('Bahadur', 'Jayabharati')	109	0,36	0,31	358,65	358,65
('Dolphy', 'Panchito')	102	0,48	0,43	704	704
('Kijaku Ôtani', 'Suminojo Ichikawa')	101	0,86	0,63	1871,88	1871,88
('Prem Nazir', 'Thikkurisi Sukumaran Nair')	101	0,39	0,23	309,12	309,12

Figure 5: Frequent pairs



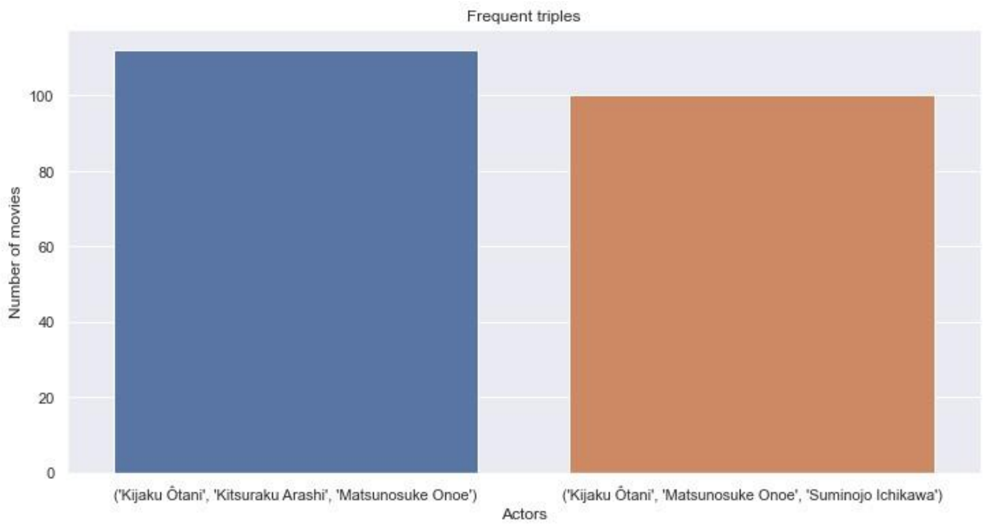
Frequent triples

The most frequent triple is a triple of Japanese actors Kijaku Ôtani, Kitsuraku Arashi and Matsunosuke Onoe. They played together 112 times. Overall, there are 2 triples of actors that are frequent and are above the threshold. Lift is far above 1 for all rules so we can say the result is stable.

Table 7: Most frequent triple

actors	joint_appearences
('Kijaku Ôtani', 'Kitsuraku Arashi', 'Matsunosuke Onoe')	112
('Kijaku Ôtani', 'Matsunosuke Onoe', 'Suminojo Ichikawa')	100

Figure 6: Most frequent triples


Frequent triples
Number of movies
('Kijaku Ôtani', 'Kitsuraku Arashi', 'Matsunosuke Onoe')
('Kijaku Ôtani', 'Matsunosuke Onoe', 'Suminojo Ichikawa')
Actors