Algorithms for Massive Data

Project 2: Market-Basket Analysis IMDB dataset

Supervised By : Prof. Dario Malchiodi

Student Name: Remon Rezk

Matriculation number: 942121

Masters Program: Data Science & Economics (DSE)

August, 2021

Contents

1 Declaration **3**

2 Abstract **4**

3 Introduction **5**
3.1 Problem Statement 5
3.2 Dataset Context & Content 5

4 Project Objective: **7**

5 Data Exploration **8**
5.1 **How is IMDB dataset Look Like?:** 8
5.2 Let's Build a Master Data Table 9
5.3 Master Data Table Visualizations 9

6 Algorithms Implementation **12**
6.1 A-Priori Algorithm Implementation 12
6.2 FP-Growth Algorithm Implementation 13
6.3 N-Gram Algorithm Implementation 15

7 Scalability & Performance **16**

8 Conclusion **17**

1 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.“

2 Abstract

In this project, The task is to implement a system finding frequent itemsets (aka market-basket analysis), analyzing of the IMDB datasets described below.

The IMDB dataset is published on Kaggle, under IMDb non-commercial licensing.

The market basket analysis consists in finding the frequent itemsets, which are items appearing together in the same baskets a sufficiently large number of times defined in advance. For this analysis, the movies are considered as baskets and the actors as items.

The algorithms used are the **Apriori**, which works by generating candidate itemsets formed by frequent items and then matching them against the dataset and the **FP-Growth**, which instead makes use of a compressed tree-like structure and it develops an efficient mining method and the N-Grams Algorithms.

The performance of the three in dealing with the task at hand is compared. In addition to the frequent itemsets, the association rules are retrieved.

3 Introduction

3.1 Problem Statement

The problem statement is to implement a system finding frequent itemsets (aka market-basket analysis), analyzing The IMDB dataset that published on Kaggle, under IMDb non-commercial licensing..

3.2 Dataset Context & Content

Dataset Context:
Each dataset is contained in a gzipped, tab-separated-values (TSV) formatted file in the UTF-8 character set. The first line in each file contains headers that describe what is in each column.

A is used to denote that a particular field is missing or null for that title/name.

Dataset Content:
title.akas.tsv.gz - Contains the following information for titles:

1- titleId (string) - a tconst, an alphanumeric unique identifier of the title.
2- ordering (integer) – a number to uniquely identify rows for a given titleId.
3- title (string) – the localized title.
4- region (string) - the region for this version of the title.
5- language (string) - the language of the title.
6- types (array) - Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd", "festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning.
7- attributes (array) - Additional terms to describe this alternative title, not enumerated.
8- isOriginalTitle (boolean) – 0: not original title; 1: original title.

titleId	ordering	title	region	language	types	attributes	isOriginalTitle
tt0000001	1	Carmencita - spanyol tánc	HU	\N	imdbDisplay	\N	0
tt0000001	2	Καρμενσίτα	GR	\N	\N	\N	0
tt0000001	3	Карменсита	RU	\N	\N	\N	0
tt0000001	4	Carmencita	US	\N	\N	\N	0
tt0000001	5	Carmencita	\N	\N	original	\N	1

only showing top 5 rows

title.basics.tsv.gz - Contains the following information for titles:

1- tconst (string) - alphanumeric unique identifier of the title.
2- titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc).
3- primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release.
4- originalTitle (string) - original title, in the original language.
5- isAdult (boolean) - 0: non-adult title; 1: adult title.
6- startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year.
7- endYear (YYYY) – TV Series end year. for all other title types.
8- runtimeMinutes – primary runtime of the title, in minutes.
9- genres (string array) – includes up to three genres associated with the title.

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

only showing top 5 rows

title.principals.tsv.gz – Contains the principal cast/crew for titles:

1- tconst (string) - alphanumeric unique identifier of the title.
2- ordering (integer) – a number to uniquely identify rows for a given titleId.
3- nconst (string) - alphanumeric unique identifier of the name/person.
4- category (string) - the category of job that person was in.
5- job (string) - the specific job title if applicable, else.
6- characters (string) - the name of the character played if applicable, else.

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

only showing top 5 rows

title.ratings.tsv.gz – Contains the IMDb rating and votes information for titles:

1- tconst (string) - alphanumeric unique identifier of the title.
2- averageRating – weighted average of all the individual user ratings.
3- numVotes - number of votes the title has received.

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

name.basics.tsv.gz – Contains the following information for names:

1- nconst (string) - alphanumeric unique identifier of the name/person.
2- primaryName (string)– name by which the person is most often credited.
3- birthYear – in YYYY format.
4- deathYear – in YYYY format if applicable, else .
5- primaryProfession (array of strings)– the top-3 professions of the person.
6- knownForTitles (array of tconsts) – titles the person is known for.

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

only showing top 5 rows

4 Project Objective:

The goal of the analysis is to implement a system finding frequent itemsets.

5 Data Exploration

5.1 How is IMDB dataset Look Like?:

IMDB datase consists from 5 datasets are: .

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

titleId	ordering	title	region	language	types	attributes	isOriginalTitle
tt0000001	1	Carmencita - span...	HU	\N	imdbDisplay	\N	0
tt0000001	2	Καρμενσίτα	GR	\N	\N	\N	0
tt0000001	3	Карменсита	RU	\N	\N	\N	0
tt0000001	4	Carmencita	US	\N	\N	\N	0
tt0000001	5	Carmencita	\N	\N	original	\N	1

only showing top 5 rows

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

only showing top 5 rows

1- Total number of rows for **names** data file: 9706922
2- Total number of rows for **title_akas** data file 19527971
3- Total number of rows for **title_basics** data file: 6321302

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

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

4- Total number of rows for **title_principles** data file: 36468817
5- Total number of rows for **title_ratings** data file: 993153

5.2 Let's Build a Master Data Table

To create some analysis & visualization on the dataset as below:
Here below how the master data table looks like:

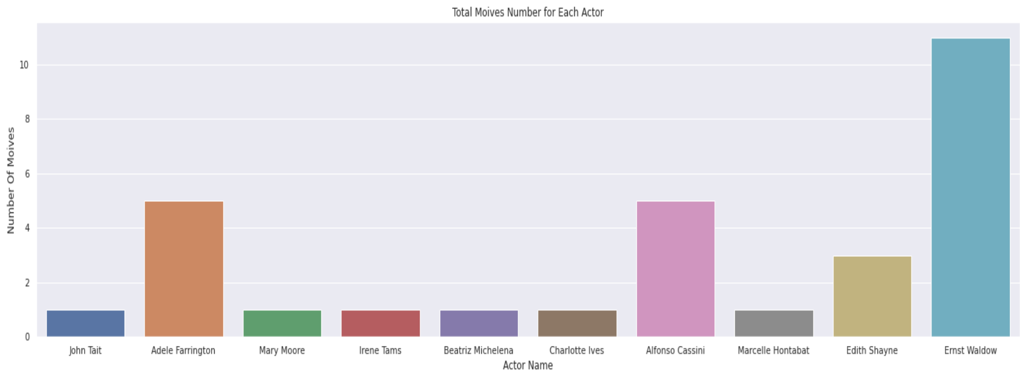
movie_id	movie_name	actor_id	actor_name	primaryProfession	movie_name	titleType	category	averageRating	numVotes	knownForTitles	startYear	genres	isAd
tt0000001	Carmencita	nm0005690	William K.L. Dickson	cinematographer,d...	Carmencita	short	director	5.6	1550	tt1496763,tt14284...	1894	Documentary,Short	
tt0000001	Carmencita	nm1588970	Carmencita	soundtrack	Carmencita	short	self	5.6	1550	tt0057728,tt0000001	1894	Documentary,Short	
tt0000001	Carmencita	nm0374658	William Heise	cinematographer,d...	Carmencita	short	cinematographer	5.6	1550	tt0241393,tt02296...	1894	Documentary,Short	
tt0000002	Le clown et ses c...	nm0721526	Émile Reynaud	director	Le clown et ses c...	short	director	6.1	186	tt0413219,tt00000...	1892	Animation,Short	
tt0000002	Le clown et ses c...	nm1335271	Gaston Paulin	composer	Le clown et ses c...	short	composer	6.1	186	tt0000003,tt21842...	1892	Animation,Short	

nly showing top 5 rows

5.3 Master Data Table Visualizations

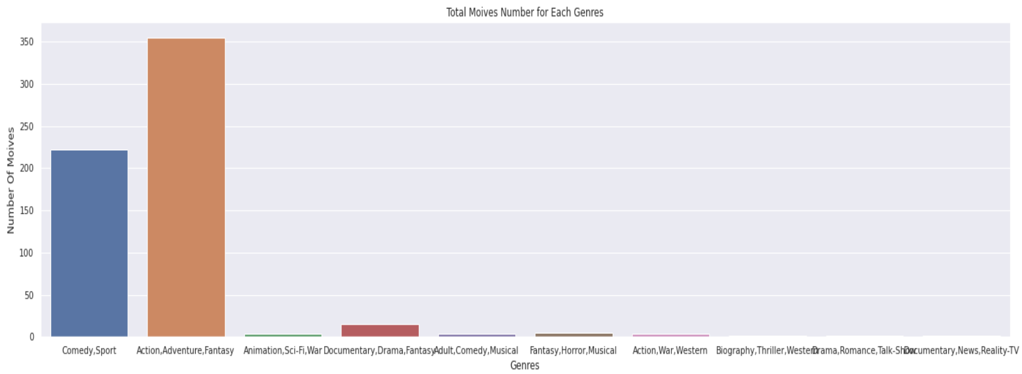
Here below we try to analysis and explore the dataset and get some insights that help us to better understand IMDB dataset

* These are the top 10 actors participated in movies.Found that "Ernst Waldow" is the top after him the American actress "Adele Farrington"



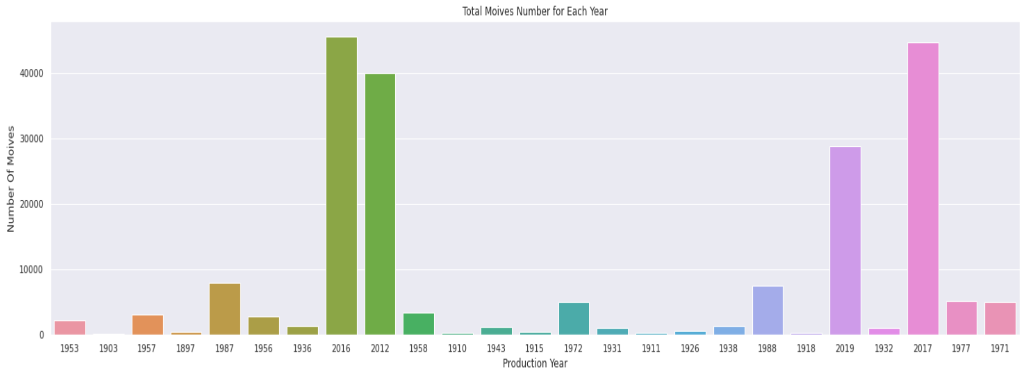
Total Movies Number for Each Actor

* These are the top 10 movies genres. Found that "Action, Adventure Fantasy" is the top genres watched after that the "Comedy Sport" movies.



Total Movies Number for Each Genres

* These are the top 25 years that movies was produced in. Found that the top years are 2016 2017.

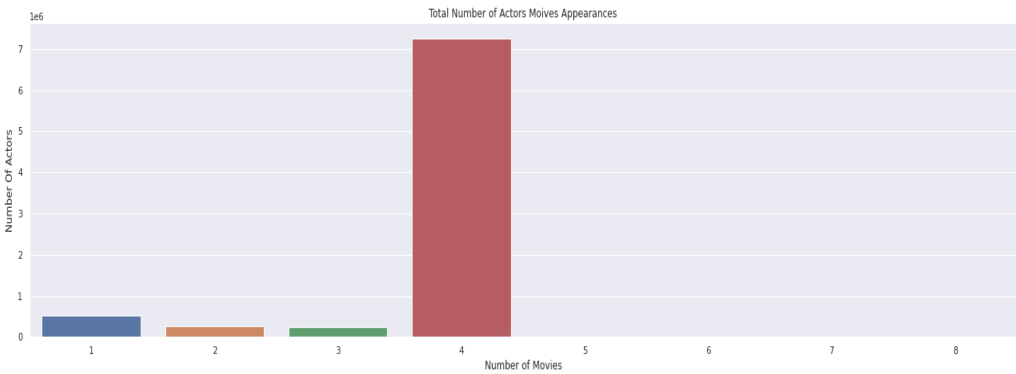


Total Movies Number Produced Each Year

* These are the most frequency for actors participated in movies. Found that most of actors participated in 4 movies.

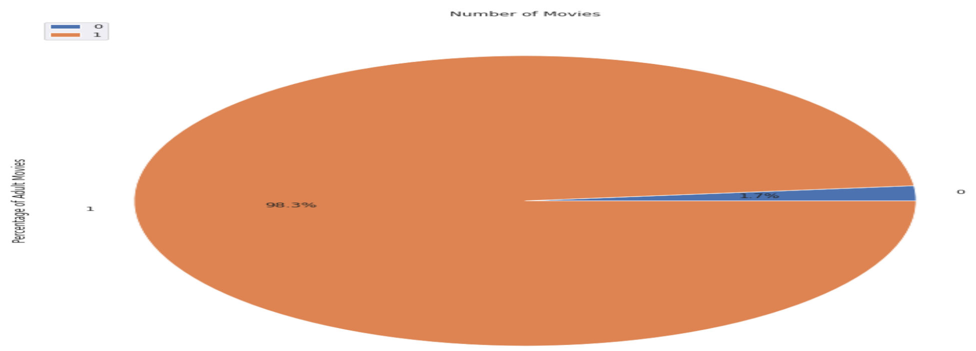
ApperanceInMovies	NumberOfActors
1	518659
2	262033
3	241405
4	7260206
5	110
6	16
7	2
8	8

Total Number of Movies that Actor Participated



Total Movies That Actors Played

* Here the percentage of Adult movies vs non adult movie. Found that Adult movies percentage is 98.3 %



Adult Movies Percentage

6 Algorithms Implementation

6.1 A-Priori Algorithm Implementation

The A-Priori algorithm is the basic implementation for finding frequent itemsets and it exploits the anti-monotone property, which says that the support of an itemset is never greater than the support of its subsets.

It takes the input file of transactions/baskets and a minimum support value, and returns a list of frequent itemsets, with the respective frequency. This Algorithm follows two main steps to reduce the search space iterative until the most frequent itemset is achieved:

1. In the first iteration of the algorithm, each item is taken as a 1-itemsets candidate, more precisely the algorithm finds frequencies by considering how many times the items occur in the data-set. It depends on the frequencies of the itemset: frequencies and support value, are obtained for every single item, by extracting every item in RDDs and calculating each unique item's frequency.

col	actor_name	actor_id
tt0001170	[Clara Williams]	[nm0930290]
tt0002591	[Martha Angerstei...	[nm0029806]
tt0003689	[Mrs. E. Walton]	[nm0910564]
tt0004272	[Eva Thatcher]	[nm0857118]
tt0004336	[Mae Tinee, Lila ...	[nm1754381, nm015...
tt0004599	[Lucille Bolton]	[nm0093420]
tt0005605	[Minnie Berlin]	[nm0075601]
tt0006204	[Blanche Bennett,...	[nm0071618, nm050...
tt0006207	[Francesca Cappel...	[nm0135474]
tt0006587	[Miriam Shelby]	[nm0790977]
tt0006819	[Alice Hechy, Rut...	[nm0372991, nm065...
tt0007011	[Mary Bunting, Vi...	[nm0120577, nm092...
tt0007565	[Hilda Moore]	[nm0601280]
tt0007694	[Pinna Nesbit, In...	[nm0626341, nm078...
tt0008160	[Nancy Caswell]	[nm0145776]
tt0008407	[Olive Bruce, Vir...	[nm0115560, nm071...
tt0008661	[Tasya Borman]	[nm1596890]
tt0008785	[Justine Cutting]	[nm0194020]
tt0008893	[Blanche Hines]	[nm0385638]
tt0009455	[Hazel Washburn]	[nm0913354]

only showing top 20 rows

2. In the second step, the algorithm counts all the candidates that consist of frequent items and checks which have counts that are equal to or greater than the support threshold. If the candidates do not meet the minimum support, then they are regarded as infrequent and thus removed.

col	actor_name	actor_id	NumberActors
tt0001170	[Clara Williams]	[nm0930290]	1
tt0002591	[Martha Angerstei...	[nm0029806]	1
tt0003689	[Mrs. E. Walton]	[nm0910564]	1
tt0004272	[Eva Thatcher]	[nm0857118]	1
tt0004336	[Mae Tinee, Lila ...	[nm1754381, nm015...	6
tt0004599	[Lucille Bolton]	[nm0093420]	1
tt0005605	[Minnie Berlin]	[nm0075601]	1
tt0006204	[Blanche Bennett,...	[nm0071618, nm050...	3
tt0006207	[Francesca Cappel...	[nm0135474]	1
tt0006587	[Miriam Shelby]	[nm0790977]	1
tt0006819	[Alice Hechy, Rut...	[nm0372991, nm065...	4
tt0007011	[Mary Bunting, Vi...	[nm0120577, nm092...	2
tt0007565	[Hilda Moore]	[nm0601280]	1
tt0007694	[Pinna Nesbit, In...	[nm0626341, nm078...	2
tt0008160	[Nancy Caswell]	[nm0145776]	1
tt0008407	[Olive Bruce, Vir...	[nm0115560, nm071...	3
tt0008661	[Tasya Borman]	[nm1596890]	1
tt0008785	[Justine Cutting]	[nm0194020]	1
tt0008893	[Blanche Hines]	[nm0385638]	1
tt0009455	[Hazel Washburn]	[nm0913354]	1

only showing top 20 rows

A-Priori Algorithm Output

```
[('nm1948317', 5),
 ('nm5756622', 5),
 ('nm2838205', 5),
 ('nm0891937', 5),
 ('nm5457566', 5),
 ('nm0686428', 5),
 ('nm1708736', 5),
 ('nm3304752', 5),
 ('nm0472407', 6),
 ('nm5613952', 5),
 ('nm0361727', 5),
 ('nm1163810', 5)]
```

6.2 FP-Growth Algorithm Implementation

FP-Growth is an algorithm which is available in Spark library and it's a good advanced alternative with respect to the classic A-priori algorithm:

1- In the first step of FP-Growth is to calculate item frequencies and identify frequent items. Different from A-priori like algorithms designed for the same purpose.

2- In the second step of FP-Growth uses a suffix tree (FP-tree) structure without generating candidate sets explicitly, which are usually expensive to generate with large datasets.

Concerning our implementation, we decided to run FP-Growth algorithm over the entire dataset considering 0.0001 as support threshold.

```
+--------------------+----+
|items               |freq|
+--------------------+----+
|[Brahmanandam]      |412 |
|[Mohanlal]          |321 |
|[Mammootty]         |312 |
|[Mithun Chakraborty]|298 |
|[Cüneyt Arkin]      |281 |
|[Sridevi]           |242 |
|[Dharmendra]        |237 |
|[Ron Jeremy]        |228 |
|[Tom Byron]         |227 |
|[Jagathi Sreekumar] |218 |
+--------------------+----+
only showing top 10 rows
```

Displaying of frequent itemsets, association rules and predictions:

items	freq
[Brahmanandam]	412
[Mohanlal]	321
[Mammootty]	312
[Mithun Chakraborty]	298
[Cüneyt Arkin]	281
[Sridevi]	242
[Dharmendra]	237
[Ron Jeremy]	228
[Tom Byron]	227
[Jagathi Sreekumar]	218

only showing top 10 rows

antecedent	consequent	confidence	lift	support
[Ric Lutze]	[Rene Bond]	0.4492753623188406	1688.140350877193	1.4474077394292545E-4
[Jayalalitha J]	[M.G. Ramachandran]	0.3	676.3452631578947	1.1205737337516809E-4
[Jayalalitha J]	[Nagesh]	0.3375	463.3615384615385	1.2606454504706641E-4
[Ray Corrigan]	[Max Terhune]	0.8723404255319149	3336.3282674772036	1.9143134618257882E-4

only showing top 4 rows

movie_id	actor_id	actor_name	prediction
tt0004272	[nm0092665, nm036...	[Wilbur Higby, Fr...	[]
tt0004336	[nm0268437, nm081...	[Marguerite Snow,...	[]
tt0005209	[nm0593671, nm039...	[DeWolf Hopper Sr...	[]
tt0006204	[nm0071601, nm007...	[George Periolat,...	[]
tt0006489	[nm0548402, nm019...	[Dorothy Dalton, ...	[]
tt0006819	[nm0435229, nm074...	[Max Ruhbeck, Eri...	[]
tt0010060	[nm0940437, nm019...	[Bert Woodruff, R...	[]
tt0011011	[nm0902615, nm051...	[Ica von Lenkeffy...	[]
tt0011031	[nm0689160, nm008...	[Paul Biensfeldt,...	[]
tt0013224	[nm0624735, nm055...	[Roy Atwell, Jame...	[]
tt0014617	[nm0435229, nm089...	[Emil Rameau, Han...	[]
tt0016361	[nm0570006, nm055...	[Malcolm McGregor...	[]
tt0016395	[nm0380965, nm017...	[Ronald Colman, J...	[]
tt0016679	[nm0363545, nm026...	[Shannon Day, Pat...	[]
tt0016814	[nm0109547, nm072...	[Elga Brink, Walt...	[]
tt0017866	[nm0602905, nm021...	[Reginald Denny, ...	[]
tt0018526	[nm0107574, nm046...	[Fred Kohler, Cli...	[]
tt0018537	[nm0234670, nm026...	[Kate Fabian, Pet...	[]
tt0019388	[nm0396768, nm011...	[Josephine Dunn, ...	[]
tt0019473	[nm0176971, nm043...	[Margaret Livings...	[]

only showing top 20 rows

items	freq
[Balkrishna, Rajkumar]	74
[Ciccio Ingrassia, Franco Franchi]	72
[Leo Gorcey, Huntz Hall]	59
[Ali, Brahmanandam]	52
[Kiyoshi Atsumi, Chieko Baishô]	52
[Gene Autry, Smiley Burnette]	50
[Peter North, Tom Byron]	49
[Nedumudi Venu, Mohanlal]	47
[Trigger, Roy Rogers]	47
[M.G. Ramachandran, M.N. Nambiar]	47
[Raymond Hatton, Johnny Mack Brown]	45
[Ashwath, Rajkumar]	42
[Roy Rogers, George 'Gabby' Hayes]	41
[Ray Corrigan, Max Terhune]	41
[Mammootty, Mohanlal]	40
[Leelavathi, Rajkumar]	40
[Udaya Kumar, Rajkumar]	40
[Mohan Babu, Brahmanandam]	39
[Aachi Manorama, Shivaji Ganesan]	39
[Nagesh, Shivaji Ganesan]	39

only showing top 20 rows

antecedent (if)	consequent (then)	confidence
[Nobuyo Ōyama, Kaneta Kimotsuki]	[Noriko Ohara]	1.0
[Black Jack]	[Allan Lane]	1.0
[Larry Simms]	[Penny Singleton]	1.0
[Larry Simms]	[Arthur Lake]	1.0
[Champion]	[Gene Autry]	1.0
[Nobuyo Ōyama, Michiko Nomura]	[Noriko Ohara]	1.0
[Larry Simms, Penny Singleton]	[Arthur Lake]	1.0
[Gabriel Dell, Leo Gorcey]	[Huntz Hall]	1.0
[Michiko Nomura, Kaneta Kimotsuki]	[Noriko Ohara]	1.0
[Penny Singleton, Arthur Lake]	[Larry Simms]	1.0
[Larry Simms, Arthur Lake]	[Penny Singleton]	1.0
[Donald F. Glut]	[G. Larry Butler]	1.0
[Dale Evans, Trigger]	[Roy Rogers]	1.0
[Trigger]	[Roy Rogers]	1.0
[Stan Laurel]	[Oliver Hardy]	1.0
[Dale Evans, Roy Rogers]	[Trigger]	1.0
[Eddy Waller, Black Jack]	[Allan Lane]	1.0
[Lou Costello]	[Bud Abbott]	0.9722222222222222
[Bud Abbott]	[Lou Costello]	0.9722222222222222
[Nobuyo Ōyama]	[Noriko Ohara]	0.96

6.3 N-Gram Algorithm Implementation

The N-Gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n1)–order Markov model.

Two benefits of N-Gram models are simplicity and scalability – with larger n, a model can store more context with a well-understood space–time trade off, enabling small experiments to scale up efficiently.

N-Gram used bigrams and counted the frequency of bigrams. A bigram is a sequence of two adjacent elements from a string of tokens. In our case, a bigram will be the names of two actors that are in a movie one after another.

movie_id	actor_id	actor_name	bigrams
tt0004272	[nm0092665, nm036...	[Wilbur Higby, Fr...	[Wilbur Higby Fra...
tt0004336	[nm0268437, nm081...	[Marguerite Snow,...	[Marguerite Snow ...
tt0005209	[nm0593671, nm039...	[DeWolf Hopper Sr...	[DeWolf Hopper Sr...
tt0006204	[nm0071601, nm007...	[George Periolat,...	[George Periolat ...
tt0006489	[nm0548402, nm019...	[Dorothy Dalton, ...	[Dorothy Dalton W...
tt0006819	[nm0435229, nm074...	[Max Ruhbeck, Eri...	[Max Ruhbeck Eric...
tt0010060	[nm0940437, nm019...	[Bert Woodruff, R...	[Bert Woodruff Ru...
tt0011011	[nm0902615, nm051...	[Ica von Lenkeffy...	[Ica von Lenkeffy...
tt0011031	[nm0689160, nm008...	[Paul Biensfeldt,...	[Paul Biensfeldt ...
tt0013224	[nm0624735, nm055...	[Roy Atwell, Jame...	[Roy Atwell James...
tt0014617	[nm0435229, nm089...	[Emil Rameau, Han...	[Emil Rameau Hans...
tt0016361	[nm0570006, nm055...	[Malcolm McGregor...	[Malcolm McGregor...
tt0016395	[nm0380965, nm017...	[Ronald Colman, J...	[Ronald Colman Je...
tt0016679	[nm0363545, nm026...	[Shannon Day, Pat...	[Shannon Day Pat ...
tt0016814	[nm0109547, nm072...	[Elga Brink, Walt...	[Elga Brink Walte...
tt0017866	[nm0602905, nm021...	[Reginald Denny, ...	[Reginald Denny C...
tt0018526	[nm0107574, nm046...	[Fred Kohler, Cli...	[Fred Kohler Cliv...
tt0018537	[nm0234670, nm026...	[Kate Fabian, Pet...	[Kate Fabian Petr...
tt0019388	[nm0396768, nm011...	[Josephine Dunn, ...	[Josephine Dunn A...
tt0019473	[nm0176971, nm043...	[Margaret Livings...	[Margaret Livings...

only showing top 20 rows

The Output of N-Grams Algorithm:

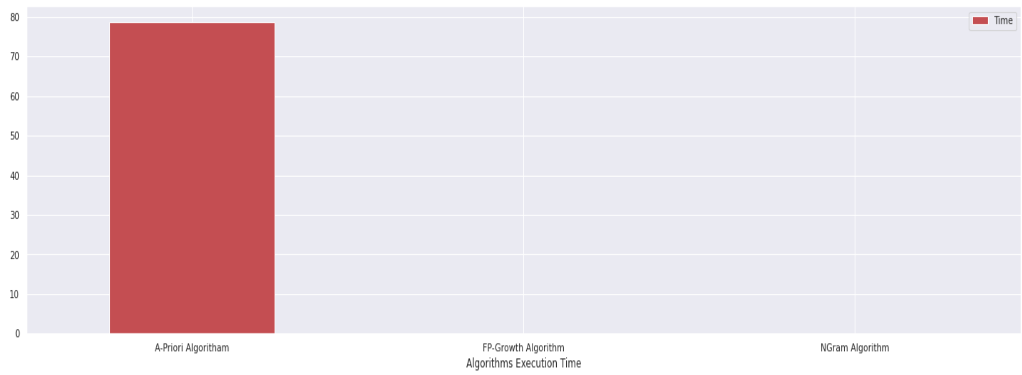
```
Streaming output truncated to the last 5000 lines.
Mercedes & Cabral : 18
Mercedes & Carreño : 12
Mercedes & Sampietro : 29
Mercedes & Carreras : 11
Caio & Blat : 15
Dirk & Benedict : 14
Dirk & Bogarde : 53
Nia & Long : 18
Pejman & Bazeghi : 25
Aleksandar & Bercek : 26
Aleksandar & Gavric : 13
Kaushik & Ganguly : 12
Kaushik & Banerjee : 18
Laurent & Grévill : 14
Laurent & Terzieff : 27
Laurent & Lucas : 14
Laurent & Lafitte : 18
Laurent & Malet : 14
Firat & Tanis : 16
Virginie & Caillat : 15
Virginie & Ledoyen : 28
Virginie & Efira : 21
Gil & Lagardère : 12
Gil & Bellows : 27
Aparna & Balamurali : 13
Aparna & Sen : 39
Asif & Ali : 54
Asif & Khan : 11
Sreenivasan & Jayaram : 12
Mariya & Vinogradova : 12
Casper & Van : 27
Stacey & Dash : 15
Stacey & Donovan : 29
Álvaro & de : 17
Vanessa & Redgrave : 52
```

7 Scalability & Performance

The performance of the three algorithms has been evaluated in terms of execution time. Due to the enormous computational cost in the generation of frequent itemsets as shown below.

FP-Growth and N-GRAMS algorithms are more scalability, simplicity and efficiently with larger dataset that A-Prior algorithm.

```
A-Prior Algorithm Running Time: 78.7818248271942l
FP-Growth Algorithm Running Time: 3.814697265625e-05
NGram Algorithm Running Time: 4.1961669921875e-05
```



Apriori Algorithm vs. FP-Growth Algorithm vs N-Gram Algorithm

8 Conclusion

The purpose of this project is to finding frequent sets of items appearing in many of the same baskets, has been accomplished.

Firstly, the absolute number of films that contain a particular set of actors/actresses has been retrieved from the chosen IMBD dataset.

Then, A-Priori, FP-Growth, N-Grams algorithms have been implemented to achieve this goal. In fact, results demonstrate how many and which actors and actresses appear more frequently in which movie. It is possible to conclude that both algorithms have reached the initial purpose, demonstrating how many and which actors and actresses appear more frequently individually and together in which films.

At the end we can say that the A-Priori algorithm implemented but requires a more execution time than the FP-Growth and N-Gram Algorithms. .