Algorithms for massive datasets

Gianluca Lo Vecchio

1 Introduction

The aim of the paper is a report on algorithms used to discover Frequent Itemsets in a dataset. The Algorithms used are Apriri, SON and FP, the dataset over which the algorithm were used include information about films, actors and crews and ratings. In particular it is to discover couples of actors, the items of our analysis, that frequently work together, thus movies are our baskets.

1.1 Key concepts

To be more precise is necessary to establish few concepts and how they interact with one another. The market-basket model is used to describe many-to-many relationships[1] in a set of data. There are two type of objects linked by this relationships, the **item** and the basket[2], the item is the object we are interested in and there can be many of it, while the basket is an event, a way to get set of items. The number of baskets is strictly superior to the number of items, since can exists potentially many combination of given set of items, not taking into account the ordering. A frequent Itemsets is defined on being supported by a certainty quantity of baskets, this quantity is called threshold. There is hierarchy of frequent Itemsets, based of the number of items presents in the Itemsets, starting from the bottom the first is the singleton, then the doubleton, the triple and so on. Naturally the quantity of singleton is major than the quantity of doubleton, and the same can be said for any comparison between classes. This is due to the logical necessity that to be a frequent itemsets is necessary to have all item in the set be frequent accordingly to every **step**, or subset, that comes before in the hierarchy. Thus a doubleton to be frequent has to supported by the threshold and thus also its singletons have to supported by it too, this is the **Monotonicity of the Itemsets**. In the same way a triple to be frequent must be composed

[1] A relationship where both type of objects can be linked to more than one element from the other class.

[2] It is also known as transaction.

of frequent couples and frequent singletons. If we instead run the process in the opposite direction from the bottom to the top of the hierarchy the highest existing frequent Itemsets is called maximal. This hard condition majorly reduce the number of Itemsets of increasing size, leaving only the truly frequent itemsets as information. The information extracted through the Frequent Itemsets are used to build association rules, finalizing the notion of likely by confidence of a specific **rule**. Basically a ratio between all of the baskets and the set that presents a feature we are interested in, like a specific couple of items. The idea behind is to construct a causation relationships, a **rule**, that explain the presence or the absence of a specific a element through the presence of one or more different elements. To achieve this the support threshold has to be reasonably high, otherwise there will be many Itemsets that have no solid and useful relationship behind it.

1.2 Algorithms

A-Priori is an algorithm that identify couples and counts the frequency of these couples, is the frequency is above the support threshold is consider and classified as frequent otherwise is ignored. If the amount of existing couples is above the real possibility of the hardware the consequence is an OOM error, and require other strategy to be solve, that respect the memory constraint. To identify and counts couples without encountering an OOM error are required two passes, during the first pass two tables are created, one convert item names to integers, the other table is a simple frequency array. Initially every new item is a new row in the two tables, a new integer to identify the items and and a 1 to represents the first occurrence. Between the two passes counts on the frequency tables are used to determine singletons and two new tables with the same purpose as before are generate from the singletons list, since based on monotonicity only among singletons is possible to find frequent pairs. During the second pass pairs are counted, and the pairs supported by the threshold are classified as frequent. The count process of the second phase required to consider only the singletons inside each baskets, generate all pairs inside the baskets and count them. Is possible to identify a list of candidates and which ones are actually significant filter through counting to the frequent singleton or pair, based of the pass. The A-Priori shows two shortcomings, the candidate generation process depends on the number of items, which leads to large candidate lists in the case of huge datasets and multiple passes are a considerable cost.

FP-Algorithm overcomes this issue, by avoiding all together the use of candidate lists, and thus more than two scans are no longer required, overcome the second phase of the algorithm. FP starts as A-Priori by scanning and defining the singletons present, then a FP tree is build with the previous information. The procedure create a root. which is null, then a basket is examined and branch is

build with using Itemsets in descedending order by frequency. The procedure is repeated for each basket, if an Itemset is common between more branch, a prefix is shared to the root to represents the link between branches.

SON algorithm process in parallel various chunks of the dataset, operating A-priori over small size chunks and integrating the results. To achieve these are required two procedures, mapping and reduction, these procedures are done in sequence two times respectively. The first round gets the list of candidate itemsets from a map function and a reduction. The second round is a filtering procedure, mapping the output of the first round and count the number of instances proportionate to the data chunk assigned, then gathering only the supported Itemsets from the candidate list. The final reduction procedure join the results from the various parallel workers summing the frequency and filtering according to the threshold, the result is a complete list of frequent Itemsets. SON algorithm achieve this with a far better use of memory than A-priori thanks to parallelize lighter interaction.

2 The Dataset

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

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
tt0000003	1	nm0721526	director	\N	\N
tt0000003	2	nm5442194	producer	producer	\N
tt0000003	3	nm1335271	composer	\N	\N
tt0000003	4	nm5442200	editor	\N	\N
tt0000004	1	nm0721526	director	\N	\N

only showing top 10 rows

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...
nm0000006	Ingrid Bergman	1915	1982	actress,soundtrac...	tt0038109,tt00368...
nm0000007	Humphrey Bogart	1899	1957	actor,soundtrack,...	tt0043265,tt00338...
nm0000008	Marlon Brando	1924	2004	actor,soundtrack,...	tt0070849,tt00787...
nm0000009	Richard Burton	1925	1984	actor,producer,so...	tt0057877,tt00878...
nm0000010	James Cagney	1899	1986	actor,soundtrack,...	tt0035575,tt00318...

only showing top 10 rows

(a) The three datasets show movie data, crew data and cast data respectively.

The dataset used by the algorithm is a left join between a table that is a subset of the first dataset with a where condition requiring $titleType =$ "*movie*" and a table build in a similar manner on the second dataset condition over $category =$ "*actor*" **OR** $category =$ "*actress*". Then the join result is grouped by the movie's title identifier and selecting only actor identifier. Finally the dataset is transform to a RDD, Resilient Distributed

tconst	nconst
tt0002591	[nm0029806, nm050...
tt0003689	[nm0910564, nm052...
tt0004272	[nm0368875, nm009...
tt0004336	[nm0268437, nm081...
tt0005209	[nm0593671, nm039...

only showing top 5 rows

Figure 2: Complete processed dataset.

Dataset. This transformation gives the possibility to divide into chunks, with redundancy, the original dataset so to exploit parallel computing though spark cluster's nodes.

3 A-Priori

The A-Priori used in the paper is build using natives Spark functions applicable to RDDs. with FlatMap a list is created, which over we apply a reducebykey and a filter, that drop items that are not sufficiently frequent accordingly to the threshold. The threshold is a function, the product, of the number of rows and a parameter (0.0003). The singletons are mapped with the map function to get an iterable that is used to obtain a list of possible frequent couples derived from singletons and then repeating the procedure used to obtain singletons over the candidate table. The Algorithm require 24 seconds. And the most frequent couple is composed of Prem Nazir and Adoor Bhasi.

Pairs of actors and actresses	Number of movies
{nm0623427, nm000...	237
{nm0006982, nm061...	122
{nm0006982, nm041...	162
{nm0046850, nm000...	169
{nm2082516, nm064...	147
{nm2373718, nm064...	126

Figure 3: A-Priori output.

4 FP-Growth

FP growth algorithm is applied using pyspark library so custom function are not required, is sufficient indicating support and Item column. Only 14 seconds are required to the algorithm, using the same parameter fro the support as in the A-Priori, 0.0003. The most frequent couple is the same as before Prem Nazir and Adoor Bhasi. The advantage of FP is the ability to produce all the Itemsets present without requiring multiple scan for every size the algorithm is looking for, but we extract with a query a size equal to two.

items	freq
[nm0623427, nm000...	237
[nm0046850, nm000...	169
[nm0419653, nm000...	162
[nm2082516, nm064...	147
[nm2373718, nm064...	126
[nm0619779, nm000...	122

Figure 4: FP output.

5 SON

The SON algorithm is an improvement over A-Priori, exploit parallel computing thus the rdd is parallelize and the support is adjusted according the number of partitions. Two round are required, using two different custom functions. The first get all the possible candidates get from the various data chunks using the A-Priori over every single chunk and aggregating the result and requires 11 seconds. Then the output of the first custom function is mapped and filtered using the same spark functions as before and 0.69 seconds are necessary. Prem Nazir and Ador Bhasi make the most frequent couple according to the SON algorithm.

Pairs of actors and actresses	Number of movies
{nm0623427, nm000...	237

Figure 5: SON output.

6 Rules

Extrapolating the frequency as couple and as singletons is possible to get rules about the probability of of an actor based on another. Applying this to the most frequent couple Bhasi is present in 585 movies, while Nazir is present in 438 films, 237 of these films presents the couple. If Bhasi is present in a movie $\frac{237}{585} = 0.41$ is the probability of also having Nazir, while $\frac{237}{438} = 0.54$ represents the reverse case when Nazir is present and we get the probability of also having Bhasi.

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.