Market-basket analysis : SON Algorithm

Giovanni Buscemi

Università degli studi di Milano

1 Introduction

The market-basket model of data describe a common form of many-many relationship between two kinds of objects : items and baskets (set of items). The primary objective in this model is finding frequent-items, which are set of items that frequently appear together in the same baskets, in order to find association with high confidence. In this project, we performed a Market-basket analysis on the Letterbox dataset. Letterbox is a movie opinion-based social network . SON A-priori algoritm was implemented to identify frequent itemset of dimension k. Due to resource limitation, the algorithm was limited to k=2

2 Algorithm and method

SON A-priori algorithm is a variant of the classic A-priori that leverages the parallelism offered by Spark, mantaining consistency and correctly eliminating both false positive and false negative. Here we define the support, which is the minimum number of occurrency of each itemset to be considered frequent.

2.1 A-priori

A-priori algorithm is designed to reduce the number of pairs that must be counted, at the expense of performing two passes over data. This can be done because of the monotonicy property, which states that if an item is frequent, all it's subset must also be frequent. An itemset is frequent if the number of occurency is greatest than a support s. This property helps to reduce the number of candidate itemset by focusing only on those that have frequent subsets. The algorithm is implemented as follow :

1. **Count the frequency of individual items**: Calculate the frequency of each individual item in the dataset.
2. **Generate candidate itemsets of size 2**: Create candidate itemsets consisting of two items each.
3. **Count itemset frequencies**: Count the occurrences of each candidate itemset in the dataset.
4. **Filter itemsets with less occurrence than the support threshold**: Remove candidate itemsets that have a frequency lower than the specified support threshold.

2.2 SON

The SON algorithm utilizes MapReduce paradigm to distribute the dataset across multiple nodes. In this approach, the dataset is divided into partitions, and A-priori algorithm is applied to each partition independently using 'mapPartitions' function, which apply A-priori to each partition, exploiting the parallelism offered by Spark. Candidates are aggregated and a full pass on the data is done in order to calculate real count of candidates and eliminate false positive. The algorithm is implemented as follow :

1. **Divide baskets into partitions**: The dataset is divided into multiple partitions to be processed in parallel.
2. **Apply A-priori algorithm to each partition**: For each partition, the A-priori algorithm is applied using a support threshold of $sp = \frac{support}{numberofpartitions}$.
3. **Collect candidates from each partition**: Each candidate frequent itemset identified in the partitions is collected.
4. **Count the total occurrences of all candidates**:
   (a) **MapPartitions function**: A `MapPartitions` function is used to count the occurrences of each candidate in each basket.
   (b) **ReduceByKey function**: A `ReduceByKey` function is then applied to sum the counts from each partition.
5. **Filter frequent candidates**: A final filter is applied to identify only the frequent candidates, thus eliminating false positives.

Note that false negative are eliminated because if an item is not frequent, than it's support is less than ps in each partition. False positive are eliminated by calculating real count.

3 Experiment

3.1 Dataset and Preprocessing

The dataset used comes from Letterboxd, a social networking platform dedicated to discussing and discovering films. For our study, we used the "actors.csv" file, which contains the names of the actors and the IDs of the films they are associated with. **Preprocessing**:

1. **Map names into integer**: Map function is applied to dataset in order to map actors name into integer value using a dictionary.
2. **GroupByKey**: Actor names are aggregated to create a basket and key is excluded.

3.2 Data scaling and result

The implementation using Spark offers scalability based on the available computational resources. However, the main workload resides in the A-priori phase and largely depends on the number of itemsets to be checked. Therefore, besides

id	name
1000001	Margot Robbie
1000001	Lewis Easter
1000001	Onyemachi Ejimofor
1000001	Cameron Everitt
1000001	Luke Field-Wright
1000001	Sasha Flesch
1000001	Adam Fogarty
1000001	Mikey French
1000001	Anna-Kay Gayle
1000001	Charlie Goddard
1000001	Marlie Goddard

Fig. 1. Actor dataset

considering the dataset size, choosing an appropriate support threshold is crucial to avoid unnecessary computations.

In Google Colab, which provides a processor with 2 cores, we achieved efficient execution times by processing 1.3% of the dataset with a support threshold set to 4. This configuration leverages the limited computational resources effectively.

Using Spark locally with a processor with 12 cores, we observed comparable execution times to the Google Colab setup when processing 50% of the dataset.

Actor 1	**Actor 2**	**Confidence**
BebeDaniels	*HaroldLloyd*	0.83
HaroldLloyd	*BebeDaniels*	1.0
CiccioIngrassia	*FrancoFranchi*	1.0
FrancoFranchi	*CiccioIngrassia*	1.0
JeffBennett	*FrankWelker*	0.8
FrankWelker	*JeffBennett*	0.31
ChingMiao	*ChengKang − Yeh*	0.67
ChengKang − Yeh	*ChingMiao*	0.8
VelimirŽivojinović	*Dragomir'Gidra'Bojanić*	0.36
Dragomir'Gidra'Bojanić	*VelimirŽivojinović*	1.0
GilbertM.Anderson	*VictorPotel*	0.8
VictorPotel	*GilbertM.Anderson*	0.67
Harry'Snub'Pollard	*BebeDaniels*	0.56
BebeDaniels	*Harry'Snub'Pollard*	0.83
Harry'Snub'Pollard	*HaroldLloyd*	0.56
HaroldLloyd	*Harry'Snub'Pollard*	1.0
RoscoeArbuckle	*AlSt.John*	0.83
AlSt.John	*RoscoeArbuckle*	0.5

Fig. 2. Example of confidence result from google colab

4 Declaration

I declare that this material, which I/We 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/us or any other person for assessment on this or any other course of study