Algorithms for massive data
Market-basket analysis

Chiara Anni
28503A
chiara.anni@studenti.unimi.it

Academic Year 2023-2024

1 Introduction

This project implements the SON algorithm to perform a Market Basket analysis on a dataset containing job skills taken from LinkedIn. To deal with such a large Dataset, all the process uses the Python library *Pyspark*. Our aim is to execute an analysis on a huge number of baskets that contain a sequence of job skills. In particular, we want to end up with the most frequent itemsets that is in our dataset.

2 Data

The dataset is taken from *Kaggle* at the following link:
https://www.kaggle.com/datasets/asaniczka/1-3m-linkedin-jobs-and-skills-2024
It presents two column, but I'm only interested in the one containing *job skills*. In each row of the dataset is contained a basket with a list of several job skills that derive from a LinkedIn profile.

3 Data preprocessing

In order to manage this large set of data, I download the dataset in a *pyspark* Dataframe. During the process, I remove all the NULL values present, and then select only the column I'm interested in. Here the dataset has a length of 1294374. To guarantee a computing time around 15 minutes, I consider a randomly sampled fraction of 0.02% of the baskets of the dataset. The dataset considered now has length of 2615 baskets. I split the baskets to separate the items inside and transform all the structure in a pyspark RDD.

4 The SON Algorithm

Based on the theory of the SON algorithm studied, I realized an implementation of it for the problem of finding the most frequent itemsets for job skills on LinkedIn. The algorithm is based on the monotonicity property: if a set of items is frequent, all its subsets are frequent. In the first step, it initially do a partitioning of the dataset into a fixed number of partitions. On each of the partitions is performed the concept of the first pass of the APriori algorithm. In fact, here it counts the occurrences of each item in the corresponding partition, and filters those that appear less times than a given fixed support trashold. This trashold is derived form the trashold established for the entire dataset, dividing it for the number of partitions (and moltiplied by 2 to consider less items and reduce the time of computing). Doing this for all partitions, and joining the results, you get the candidate items to be frequent in the whole set.
As a second step, I run the APriori algorithm on the entire dataset, counting here the occurrences of the candidate items, and filtering those that are not as

frequent as the established threshold. So we found the singleton frequent in the entire dataset.

Executing all the next pass to find the most frequent couples, triplets and subsets in general is just an iteration of this procedure. In particular, I search again the candidate subsets in all the partitions, and then I count them in the entire dataset.

Obviously, our result will depend on the threshold that we establish for the dataset and for the single partitions. The code for the APriori algorithm is taken from the github of the teaching assistant Francesco Bertolotti at the following link:
https://github.com/f14-bertolotti/labs/tree/main/DSE/MarketBasket

5 Scalability

Processing large amounts of data is one of the most recent problems the world of technology is facing. The Pyspark library of Python aims to manage the problems that can arise in dealing with operations on very large datasets. This project has all the computations based on pyspark structure and functions, that are done parallelized at many nodes, and this allows scalability. So, all the project is able to be reproduced on the entire dataset.

6 Results

I run the Son algorithm with a frequency support of 52, that is the 2% of the portion of dataset I'm considering. I'm changing the number of partition considered from 6 to 10. In each of these partitions, I consider a support of 10. For all the pass I found:

- Most frequent singletons:
  ('Communication', 742)
  ('Teamwork', 423)
  ('Leadership', 381)
- Most frequent couples:
  ('Communication', 'Teamwork'), 275)
  (('Communication', 'Leadership'), 249)
  (('Communication', 'Customer service'), 150)
- Most frequent triplets:
  (('Communication', 'Leadership', 'Teamwork'), 101)
  (('Communication', 'Problem Solving', 'Teamwork'), 72)
  (('Communication', 'Customer Service', 'Teamwork'), 66)
- For the quadruple I don't found nothing that is frequent in the dataset with this support trashold, but two of the 7 candidates extracted from the partitions are:

(('Adaptability', 'Collaboration', 'Communication', 'Problem Solving'), 27)
(('Collaboration', 'Communication', 'Leadership', 'Problem Solving'), 26).

7 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, and including any code produced using generative AI systems. 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."