LINGI2364: Mining Patterns in Data
Project 1: Implementing Apriori

Siegfried Nijssen, Charles Thomas

Due October 16, 23:55

1 Context

This project is focused on the Apriori algorithm which aims to find the frequent itemsets in a dataset given a fixed minimum support. The Apriori algorithm is the most basic *join-based* algorithm for frequent itemset mining. It exploits the *anti-monotonicity* property and uses a *level-wise* approach. Many optimisations have been proposed to improve the performances of the basic implementation. In this project, you will have to implement a variation of the algorithm and compare its performances with another frequent itemset miner.

2 Directives

The project will be done by groups of two students. Each group will have to register on moodle.

For this project, you will have to implement **at least two** different frequent itemset miner algorithms in Java. One of them **must** be a version of the Aprioi algorithm. The other must be either a different version of the Apriori algorithm with one or more optimisations or an implementation of a Depth First Search algorithm such as ECLAT. You will then have to compare your frequent itemset miners on several datasets with different support values in order to determine the impact of your implementation choices and optimisation(s) on the performance of the algorithms.

A template is provided. It includes the Dataset and VerticalDataset classes that can be used to read a dataset file and access its items and transactions (see Section 4.1 for more detail). You are free to modify or not use at all this template as long as you provide an `FrequentItemsetMiner` class which contains method named `apriori` that calls your implementation of the apriori algorithm and a second method named `alternativeMiner` that calls your second implementation (Depth First or Apriori variation). These methods must have the following signatures:

```
public static void apriori(String filepath, double minFrequency)
public static void alternativeMiner(String filepath, double minFrequency)
```

- The argument `filepath` corresponds to the path to a dataset file. The format of the dataset files is detailed in Section 4.2.
- The argument `minFrequency` corresponds to the minimum frequency that an itemset must have in order to be considered as frequent. The frequency of an itemset is represented by a double and is defined as its support (the number of transactions containing an itemset) divided by the total number of transactions in the dataset.

Each method must output every frequent itemset found by your implementation of the algorithm on the standard output. Each itemset must be printed on a single line following this format: "[<item 1>, <item 2>, ... <item k>] (<frequency>)". For example, an itemset containing the items 1, 2, 3 and having a frequency of 0.92 will be printed as: "[1, 2, 3] (0.92)". Note that the items in an itemset should be ordered according to the lexicographical order but the order in which the itemsets are printed does not matter.

You will be graded based on several criteria:

- The correctness of your implementations.
- The optimisation(s) that you implemented and the performances of your most advanced solution.

- The quality, structure and readability of your code.
- The quality and relevance of your report, justifications and performance analysis.

The deadline for this project is **Monday 16th of October at 11:55 pm**. Your submission should be uploaded in a single zip archive on moodle. The archive will contain all the source files of your program and your report in pdf format.

3 Report

Your report must not exceed 4 pages. It has to contain short descriptions of your different implementations as well as the optimisation(s) that you added. You have to explain and justify your implementation choices. The report must also contain an experimental comparison of the performance in terms of time (and optionally memory) on several datasets with different support values. You can also briefly discuss the difficulties that you encountered during the project.

Your report must contain the number of your group as well as the names and NOMAs of each member. It should be written in correct English, be precise and concise. Do not hesitate to use tables or graphics to depict the results of your comparison of the variations of the algorithm.

4 Resources

4.1 Provided template

We provide a small template that can serve as starting point of your implementation. It is available on moodle (in the `template.zip` archive). It is composed of three classes:

- The `Dataset` class is a utility class that reads a dataset file and allows to access its transactions and items through dedicated methods. Note that the whole dataset content is loaded in memory for faster access during the computation of your algorithm. This will take some place in your RAM depending on the dataset used but shouldn't cause any problem with the files given for the project. For the same reason, it may also take some time to initialize the object. Take this into account in your evaluations of the performance of your algorithms.

  You are free to not use or modify this class in any way you want.

- The `VerticalDataset` class is a variant of the `Dataset` class that provides a vertical representation of the dataset.

  As with the `Dataset` class, the whole dataset is loaded in memory and you are free to not use or modify this class in any way you want.

- The `FrequentItemsetMiner` class provides a very simple CLI to launch the algorithms. It should contain the methods apriori alternativeMiner that launch your implementations.

4.2 Datasets

Six different datasets are available on moodle in the `datasets.zip` archive. Each dataset file has an extension in .dat. The files have the following format: Each line corresponds to a transaction. A transaction is represented by a series of integers separated by single spaces and sorted in ascending order which represent the different items present in the transaction. No item is included more than once in a transaction.

The `toy.dat` dataset is a small dataset that can be used to track manually the execution of your algorithms. The other datasets come from this site: `http://fimi.ua.ac.be/data/`.

We provided two files to check if your implementation:

- `toy_itemsets0125.txt` contains all the itemsets that your algorithm should find in the `toy` dataset for a minimum frequency of 0.125.
- `accidents_itemsets08.txt` contains all the itemsets that your algorithm should find in the `accidents` dataset for a minimum frequency of 0.8.

5 Some tips

- The algorithms might take a lot of time on some of the datasets for low frequency values. When testing your implementations, always start with a high frequency and decrease it until your algorithm takes too much time.
- The performance analysis of your algorithms is important and will require some time. Do not underestimate this part of the project.
- Make full use of the time allocated. Do not start the project just before the deadline!
- Do not forget to comment your code.
- Plagiarism is forbidden and will be checked against! Do not share code between groups. If you use online resources, cite them.