1. Aim of the project

Aim of the project is to prepare, implement and verify experimentally SPAM algorithm, regarding mining sequential patterns. The project should be implemented in Python language without using any of the datamining libraries available.

2. Description of the algorithm

The pseudo-code of the algorithm is presented below:

SPAM(*SDB, minsup*)

Scan *SDB* to create $V(SDB)$ and identify F_1, the list of frequent items.
FOR each item $s \in F_1$,
    SEARCH($\langle s \rangle$, F_1, $\{e \in F_1 \mid e \succ_{lex} s\}$, *minsup*).

SEARCH(*pat*, S_n, I_n, *minsup*)

Output pattern *pat*.
$S_{temp} := I_{temp} := \emptyset$
FOR each item $j \in S_n$,
    IF the s-extension of *pat* is frequent **THEN** $S_{temp} := S_{temp} \cup \{i\}$.
FOR each item $j \in S_{temp}$,
    SEARCH(the s-extension of *pat* with j, S_{temp}, $\{e \in S_{temp} \mid e \succ_{lex} j\}$, *minsup*).
FOR each item $j \in I_n$,
    IF the i-extension of *pat* is frequent **THEN** $I_{temp} := I_{temp} \cup \{i\}$.
FOR each item $j \in I_{temp}$,
    SEARCH(i-extension of *pat* with j, S_{temp}, $\{e \in I_{temp} \mid e \succ_{lex} j\}$, *minsup*).

3. Datasets used for testing

During the project two datasets have been used:

- small, generated by hand dataset with 4 sequences and 2 different items to verify proper implementation of the algorithm, as follows:
    - 1: {a}, {a,b}, {a}, {a,b}
    - 2: {a,b}, {b}
    - 3: {a,b}, {a}, {b}
    - 4. {a}, {b}

- Sign dataset, containing 731 sequences and 310 different items;

4. Results

Algorithm has been used to find sequential patterns in both datasets above:

4.1. Custom dataset

Algorithm with *minSup* set to 3 returned following frequent patterns: {a}; {b}; {a,b}; {a}, {b}; {b}, {b}; {a,b}, {b};

Verifying results by hand yields the same results (Searching for frequent patterns with *sup*>=3 implies that it has to be supported from at least one of transaction sets 2,4, which means that the only possible frequent patterns are those sequences which are subsequences of either of those two) .

4.2. Sign dataset

The SPAM algorithm launched on our main dataset has found 100 frequent sequential patterns using a support threshold equal to 400 which corresponds to the relative support of around 0.547.
Instead, considering a higher threshold (i.e. 600), we can notice how the set of results is significantly reduced to just 7 frequent sequential patterns discovered. Example mined pattern is presented below:

```
6:
Sequence: {1},{253}
Absolute support: 616
Relative support: 0.842681
```

5. Conclusions

5.1. Results analysis

Tests show that although the algorithm works for the given dataset, this specific dataset contains only itemsets with one item each, therefore the i-extension part of the algorithm remains unused and one of the strengths of it is neutralized.
Despite this, SPAM performs well and is one of the fastest sequential pattern mining algorithm

5.2. Potential uses

One of the fields that the algorithm really shines is finding a comprehensive list of all sequential patterns, which later can be processed via means of rule induction to generate sequential rules about the dataset. One can obtain vital information from a dataset this way, assuming the dataset is appropriate for finding such rules. One of the examples would be the laboratory task where one had to find out what occurrences may cause hypoglycemic symptoms to appear. One of the methods to achieve this will be described and implemented as an additional project task.

6. Sequential rule generation

6.1. Methodology

Given a list of all frequent sequences in a dataset firstly we gather all the sequences with length of at least 2. From those sequences we can generate possible *length-1* rules, e.g.:

- sequence {a},{b} generate rule {a} → {b};
- sequence {a},{b},{c} generate rules {a},{b}→{c} and {a}→{b},{c};

Of course, longer sequences can generate more rules than suggested but since all of the subsequences of a frequent sequence are also frequent, those will be included while processing the subsequences.

In the next step the potential rules will be evaluated. Since all of the subsequences of frequent sequence are also frequent, the data regarding support for them is also available - therefore we can calculate support, confidence and lift of a rule.

6.2. Results

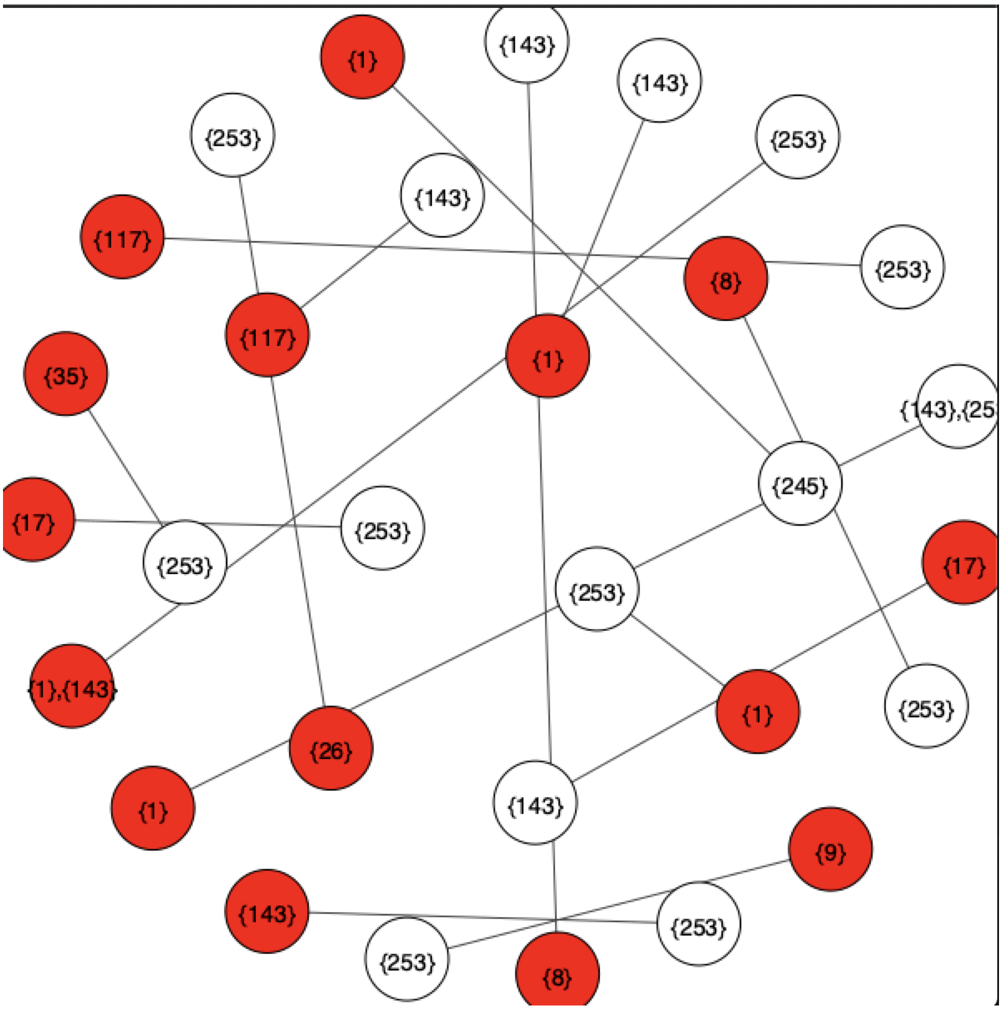
Example set of rules discovered for the Sign dataset with *minsup* = 500 is presented below:

```
Found 15 sequential rules:

Rule 1:
lh: {1}
rh: {143}
Support: 0.785226
Confidence: 0.844118
Lift: 0.907426

Rule 2:
lh: {1}
rh: {143},{253}
Support: 0.711354
Confidence: 0.764706
Lift: 0.822059
```

Those rules have been visualised using *igraph* library (red nodes represent left-hand side of the rule and white nodes represent right-hand side of the rule):



7. CM-SPAM

To improve the performance of the SPAM algorithm, an extended version of it was implemented called CM-SPAM which involves pruning the candidate searching process by using CMAP structures. To qualitatively compare approaches several tests have been performed and the execution time was measured. Each test was performed on the same sign dataset with minimal support equal to 500:

- standard SPAM algorithm approach: 5,43s
- CM-SPAM algorithm approach not counting constructing CMAP structures: 5,42s

The experiments were repeated, this time for value of minimal support of 300:

- standard SPAM algorithm approach: 9,35s
- CM-SPAM algorithm approach not counting constructing CMAP structures: 9,34s

It is clearly visible that the introduction of pruning does not improve the performance of the algorithm in a significant way. Moreover it introduces costly implementation of CMAP, even when using hashing algorithms. Perhaps the dataset experimented on does not benefit in pruning in a major way. It may be the case that elements co-occur in many sequences to the point that it is unlikely to find many pruning candidates.

In the next step the dataset used for experimentation will be modified in such a way to actually include sequences of elements of greater length than 1. To achieve that a random attribute is going to be added for every element of the dataset. Introducing random items would also make the algorithm benefit more from pruning. Results for such dataset and minsup=500 are:

- standard SPAM algorithm approach: 6,01s
- CM-SPAM algorithm approach: 5,97s

The experiment was repeated with minsup=300:

- standard SPAM algorithm approach: 14,44s
- CM-SPAM algorithm approach: 13,95s

And for minsup=200:

- standard SPAM algorithm approach: 45,54s
- CM-SPAM algorithm approach: 40,03s

The difference in all cases was not enormous, but still somewhat significant, especially regarding the fact that the algorithms may not be implemented in an optimal way, especially one regarding not perfect implementation of hash lists for CM-SPAM algorithm, which would skew the results even more in CM-SPAM favor. Also the relative difference of execution time consists of overhead of visualising results and generating rules - that is why only the raw difference matters.

Also it is visible that in general smaller minsup yields a bigger difference (at least in case of this dataset).

In the last experiment a totally random dataset of size of the previous dataset was created, then it was tested for minSup = 50:

- standard SPAM algorithm approach: 40,87s
- CM-SPAM algorithm approach: 38,71s

All the previous conclusions apply for this experiment as well.

All the experiments yold the same results in regard to sequences and rules generation

8. Final words

SPAM algorithm is used for generating frequent sequences of the dataset and does its job really well. Also such rules may be useful in a handful of related tasks such as generating sequential rules which give us knowledge about the dataset.

It is also worth noting that CM-SPAM algorithm which uses cmaps to prune candidate frequent sequences makes the algorithm even faster with no cost to quality of the results, but it does not benefit every dataset.

9. References

- http://www.philippe-fournier-viger.com/spmf/PAKDD2014_sequential_pattern_mining_CM-SPADE_CM-SPAM.pdf
- http://www.philippe-fournier-viger.com/spmf/index.php?link=datasets.php
- http://www.philippe-fournier-viger.com/spmf/datasets/SIGN.txt