Coding Assignments #1

No collaboration.

Other materials

- https://docs.python.org/3/tutorial/datastructures.html#sets
  - data structures of python
- https://docs.python.org/3/library/stdtypes.html#frozenset
  - python frozenset

How to execute the program

First, place `CA1_task1.py`, `CA1_task2.py`, `CA1_task3.py` in the `CA1_material` directory. Each python script can be executed with `python CA1_task*.py`. Python 3.9 or newer version is required to run.

Structure of scripts and short briefs

`CA1_task1.py`

This script contains typing annotations, some utility functions such as logging and the function `calc_support` to calcurate the absolute support number of itemsets. This script is imported as a module in the following scripts.

`calc_support` was tested with the plants dataset. The reference result is from `pyfim.py` in the support material of Notebook Assignments#1.

The absolute support number of a itemset `{"ri", "va", "ma", "ny"}` was 2783. The itemset was converted to frozenset of integer by using mapping dictionary `U`.

Result of `pyfim:fim`

```
>>> with open("plants/plants.data") as f:
>>>     lines = f.readlines()
>>> tracts = []
>>> for line in lines:
>>>     line = line.strip()
>>>     tracts.append(frozenset(line.split(",")[1:]))
>>>
>>> from pyfim import fim
>>> fi = fim(tracts, target="s", supp=8)
>>> result = sorted(fi, key=lambda s: s[1])[0]
>>> print(f"itemsets: {result[0]}, supp={result[1]}")
itemsets: ('ri', 'va', 'ma', 'ny'), supp=2783
```

Result of `CA1_task1:calc_support`

```
>>> from CA1_task1 import get_transactions, calc_support
>>> tracts, U = get_transactions("plants")
>>> itemset = ("ri", "va", "ma", "ny")
>>> itemset = frozenset([U.index(item) for item in itemset])
>>> support = calc_support(itemset, tracts)
>>> print(f"{support=}")
support=2783
```

CA1_task2.py

This script defines the *apriori* algorithm. The three steps of the algorithm are defined as functions, `gen_candidates`, `prune_candidates` and `get_frequent_itemsets`.

This script was tested with the transactions dataset of the exercise sheet#1. Items are converted to integers from alphabets following the mapping, d → 0, a → 1, b → 2, c → 3. The result of this script was aligned with the answers of exercise.

Result of apriori algorithm (`CA1_task2:main`)

```
>>> from CA1_task2 import main
itemsets = [0], supp = 5
itemsets = [1], supp = 4
itemsets = [2], supp = 3
itemsets = [3], supp = 2
itemsets = [0, 1], supp = 3
itemsets = [1, 2], supp = 2
itemsets = [0, 3], supp = 2
itemsets = [0, 2], supp = 3
itemsets = [0, 1, 2], supp = 2
```

CA1_task3.py

In this script, the algorithm implemented in `CA1_task2.py` is aplied to the plants dataset. Through the algorithm, total 1988 frequent itemsets were found in 41 seconds. On the other hand, `fim` of `pyfim` consumes less than 1 second to execute. A detailed analysis of the execution time by `log_ts` showed that `prune_candidates` was the slowest and it consumed 75% of the total time.

CA1_task4.py

This script implements an algorithm that mine association rules from frequent itemsets as a function `assoc_rules`. 5455 association rules are found (min_supp=4000, min_conf=0.5) in 8.913885 sec.