Report for hw1

2018000337 장호우

Environment

- Windows 10
- Python 3.7.6 64-bit

Content

Summary of your algorithm

Apriori is an algorithm for frequent item set mining and association rule learning over transactions.

At first, get all candidates from input file and create frequency map for candidates. Scan transactions and count frequency of candidates, return candidates with their frequency. Repeat this scanning process to generate all kinds of candidates util they have no longer candidates. To generate rules with confidence from frequency map. At last, calculate support and confidence then write data with format.

Detailed description of each function

To load data as list

```
def read_data(path=args.input_file):
    with open(path, 'r') as f:
        return [list(map(int, line.split())) for line in f.readlines()]
```

The main function

```
def apriori(data=read_data(), support=args.min_sup/100, k=2):
    # get all candidates from input file
    candidates = set(frozenset([i]) for i in set(chain(*data)))

    # scan transactions and count frequency of candidates, return candidates
with their frequency
    def scan_data(data=data, candidates=candidates):
        candidate_count = defaultdict(int)
        for tid in data:
            for cand in candidates:
                if cand.issubset(tid):
                    candidate_count[cand] += 1
        res = dict()
        for key, value in candidate_count.items():
            if float(value/len(data)) >= support:
                res[key] = float(value/len(data))
        return res

    # generate candidates
    def item_set(item, k):
        new_candidates = []
        for i in itemset:
            for j in itemset:
                if len(i.union(j)) == k:
                    new_candidates.append(i.union(j))
        candidates = set(new_candidates)
        return candidates

    res = dict()
    # repeat this process to generate all kinds of candidates
    while candidates:
        itemset = scan_data(candidates=candidates)
        res[k-1] = itemset
        candidates = item_set(itemset, k)
        k += 1
    return res
```

To generate rules with confidence from frequency map.

```
def rules(res):
    # get the support from dict
    def get_support(item):
        return res[len(item)][item]

    # calculate support and confidence
    for key, value in res.items():
        for item in value:
            elements = [combinations(item, i) for i, e in enumerate(item, 1)]
            for ele in map(frozenset, chain(*elements)):
                if item.difference(ele):
                    conf = get_support(item) / get_support(ele)
                    yield ele, item.difference(ele),
round(get_support(item)*100, 2), round(conf*100, 2)
```

To write data with format

```
def write_data(rules, path=args.output_file):
    with open(path, 'w') as f:
        for item, ass, support, confidence in rules:
            f.write('{:10s}\t{:10s}\t{:.2f}\t{:.2f}\n'.format(
                '{{{}}}'.format(','.join(map(str, item))),
                '{{{}}}'.format(','.join(map(str, ass))),
                support,
                confidence
            ))
```

How to run it

```
python apriori.py 5 input.txt output.txt
```