**What is MISApriori?**

MISApriori is an algorithm for mining frequent itemsets by using multiple minimum supports. It is a generalization of the Apriori algorithm, which uses a single minimum support threshold.

The idea behind MSApriori is that different minimum supports could be used to consider the fact that some items are less frequent than others in a dataset.

**What is the input of this algorithm?**

The input of **MSApriori** is a transaction database and two parameters named beta (a value between 0 and 1) and LS (a value between 0 and 1). These parameters are used to determine a minimum support for each item.

A **transaction database** is a set of transactions. Each **transaction** is a set of items. For example, consider the following transaction database. It contains 6 transactions (1, 2, ..., 5, 6) and 5 items (1, 2, 3, 4, 5). For example, the first transaction represents the set of items 1, 2, 4 and 5. This database is provided as the file **docword.xyz.txt**. It is important to note that an item is not allowed to appear twice in the same transaction and that items are assumed to be sorted by lexicographical order in a transaction. We will ignore the 3rd column in the input.

|  |  |
| --- | --- |
| **transaction** | **items** |
| 1 | 1 |
| 1 | 2 |
| 1 | 4 |
| 1 | 5 |
| 2 | 2 |
| 2 | 3 |
| 2 | 5 |
| 3 | 1 |
| 3 | 2 |
| 3 | 4 |
| 3 | 5 |
| 4 | 1 |
| 4 | 2 |
| 4 | 3 |
| 4 | 5 |
| 5 | 1 |
| 5 | 2 |
| 5 | 3 |
| 5 | 4 |
| 5 | 5 |
| 6 | 2 |
| 6 | 3 |
| 6 | 4 |

**What is the output of this algorithm?**

The output of **MSApriori**is the set of all frequent itemsets contained in the database.

Contrarily to the original Apriori algorithm, **MSApriori** use **multiple minimum supports thresholds** instead of just one. In fact, MSApriori uses a minimum support value for each item. Because it would be time consuming to set a minimum support threshold value for each item for a large database, the thresholds are determined automatically by using two user-specified parameters named beta (0 <= B <= 1) and LS (0 <= LS <= 1).

The**minimum support of an item *k*** is then defined as the greatest value between:

* LS
* and B x f(*k*) where f(*k*) is the number of transactions containing the item*k*.

Note that if B is set to 0, there will be a single minimum support for all items and this will be equivalent to the regular Apriori algorithm.

The **support of an itemset** is the number of transactions containing the itemset divided by the total number of transactions. An itemset is a frequent itemset if its support is higher or equal to the smallest minimum support threshold from the minimum support thresholds of all its items.

**Why MSApriori is useful? It is useful because it allows discovering frequent itemsets containing rare items (if their minimum support is set low).**

If we run **MSApriori** on the previous transaction database with *beta =*0.4 and *LS*= 0.2, we obtain the following result:

1 supp: 4  
2 supp: 6  
3 supp: 4  
4 supp: 4  
5 supp: 5  
1 2 Support: 4  
1 3 Support: 2  
1 4 Support: 3  
1 5 Support: 4  
2 3 Support: 4  
2 4 Support: 4  
2 5 Support: 5  
3 4 Support: 2  
3 5 Support: 3  
4 5 Support: 3  
1 2 3 Support: 2  
1 2 4 Support: 3  
1 2 5 Support: 4  
1 3 5 Support: 2  
1 4 5 Support: 3  
2 3 4 Support: 2  
2 3 5 Support: 3  
2 4 5 Support: 3  
1 2 3 5 Support: 2

1 2 4 5 Support: 3

Note that here the support is expressed by an integer value which represents the number of transactions containing the itemset. For example, itemset {2, 3 5} has a support of 3 because it appears in three transactions, namely t2, t4 and t5. This integer value can be converted as a percentage by dividing by the total number of transactions.

Input file format

The **input file format** of **MSApriori**is defined as follows. It is a text file. The file input.txt begins with 3 header lines namely number of documents in the collection, number of words whose frequency is counted (i.e., number of words in words.txt) and number of non-zero frequency entries for this collection (i.e., it is 3 less than the number of lines in input.txt).

This is followed by lines of the form

docID wordID count

where count is the number of times the word with id wordID appears in document with id docID.

An item is represented by wordID as a positive integer. Each item is separated from the following item by a space. It is assumed that items are sorted according to a total order and that no item can appear twice in the same transaction. For example, for the previous example, the input file is defined as follows:

39861  
28102  
3710420  
1 118 1  
1 285 1  
1 1229 1  
1 1688 1

2 22435 1  
2 22934 1  
2 23028 1  
2 23557 1

3 22644 1  
3 22659 1  
3 22805 1

vocab.XYZ.txt is the vocabulary file, listing all words that appear in the collection XYZ, one word per line. Each word has an implicit wordID that is its line number in this file, starting with 1 (the word on line 1 has wordID 1, the word on line 2 has wordID 2, ...)

Output file format

The **output file format** of **MSApriori**is defined as follows. It is a text file, where each line represents a **frequent itemset**. On each line, the items of the itemset are first listed. Each item is represented by an integer, followed by a single space. After, all the items, the keyword "#SUP:" appears, which is followed by an integer value indicating the support of that itemset.

1 #SUP: 4  
2 #SUP: 6  
3 #SUP: 4  
4 #SUP: 4  
5 #SUP: 5  
1 2 #SUP: 4  
1 3 #SUP: 2  
1 4 #SUP: 3  
1 5 #SUP: 4  
2 3 #SUP: 4  
2 4 #SUP: 4  
2 5 #SUP: 5  
3 4 #SUP: 2  
3 5 #SUP: 3  
4 5 #SUP: 3  
1 2 3 #SUP: 2  
1 2 4 #SUP: 3  
1 2 5 #SUP: 4  
1 3 5 #SUP: 2  
1 4 5 #SUP: 3  
2 3 4 #SUP: 2  
2 3 5 #SUP: 3  
2 4 5 #SUP: 3  
1 2 3 5 #SUP: 2  
1 2 4 5 #SUP: 3

For example, the first line indicates that the itemset {1} has a support of 4 transactions. The following lines follows the same format.

**We ran the MSApriori algorithm for docword.nips.txt, for beta =0.5 and LS=0.5. The output is attached.**

============= MSAPRIORI – STATS FOR docword.nips.txt =============

MSAPRIORI with beta= 0.5 and LS= 0.5

The algorithm stopped at level 10, because there is no candidate

Frequent itemsets count : 31582

Maximum memory usage : 1056.106460571289 mb

Total time ~ 295195 ms

**We ran the MSApriori algorithm for docword.kos.txt, for beta =0.5 and LS=0.1. The output is attached.**

============= MSAPRIORI – STATS FOR docword.kos.txt =============

MSAPRIORI with beta= 0.5 and LS= 0.1

The algorithm stopped at level 8, because there is no candidate

Frequent itemsets count : 2645

Maximum memory usage : 430.67286682128906 mb

Total time ~ 28037 ms