FAE-ad

September 5, 2022

1 Advanced Tutorial on Implementing FAE Algorithm

In this tutorial, we will discuss two approaches to find frequent patterns in big data using FAE algorithm. 1. Advanced approach: Here, we generalize the basic approach by presenting the steps to discover frequent patterns using multiple specified counte.

In this tutorial, we explain how the FAE algorithm can be implemented by varying the specified counte values

Step 1: Import the FAE algorithm and pandas data frame

[1]:
```
from PAMI.frequentPattern.topk import FAE  as alg
import pandas as pd
```

Step 2: Specify the following input parameters

[2]:
```
inputFile = 'transactional_T10I4D100K.csv'
seperator='\t'
kCountList = [100, 150, 200, 250, 300]
#minimumSupport can also specified between 0 to 1. E.g., minSupList = [0.005, 0.
↪006, 0.007, 0.008, 0.009]

result = pd.DataFrame(columns=['algorithm', 'minSup', 'patterns', 'runtime',␣
↪'memory'])
#initialize a data frame to store the results of FAE algorithm
```

Step 3: Execute the FAE algorithm using a for loop

[3]:
```
algorithm = 'FAE'  #specify the algorithm name
for minSupCount in minimumSupportCountList:
    obj = alg.FAE('transactional_T10I4D100K.csv', k=minSupCount, sep=seperator)
    obj.startMine()
    #store the results in the data frame
    result.loc[result.shape[0]] = [algorithm, minSupCount, len(obj.
↪getPatterns()), obj.getRuntime(), obj.getMemoryRSS()]
```

```
100
FAE has successfully generated top-k frequent patterns
150
FAE has successfully generated top-k frequent patterns
200
FAE has successfully generated top-k frequent patterns
250
FAE has successfully generated top-k frequent patterns
300
FAE has successfully generated top-k frequent patterns
```

[4]:
```
print(result)
```

```
  algorithm  minSup  patterns   runtime     memory
0       FAE     100       100  1.639922  216731648
1       FAE     150       150  2.565559  217001984
2       FAE     200       200  3.712384  217690112
3       FAE     250       250  4.976268  217714688
4       FAE     300       300  6.439660  217649152
```

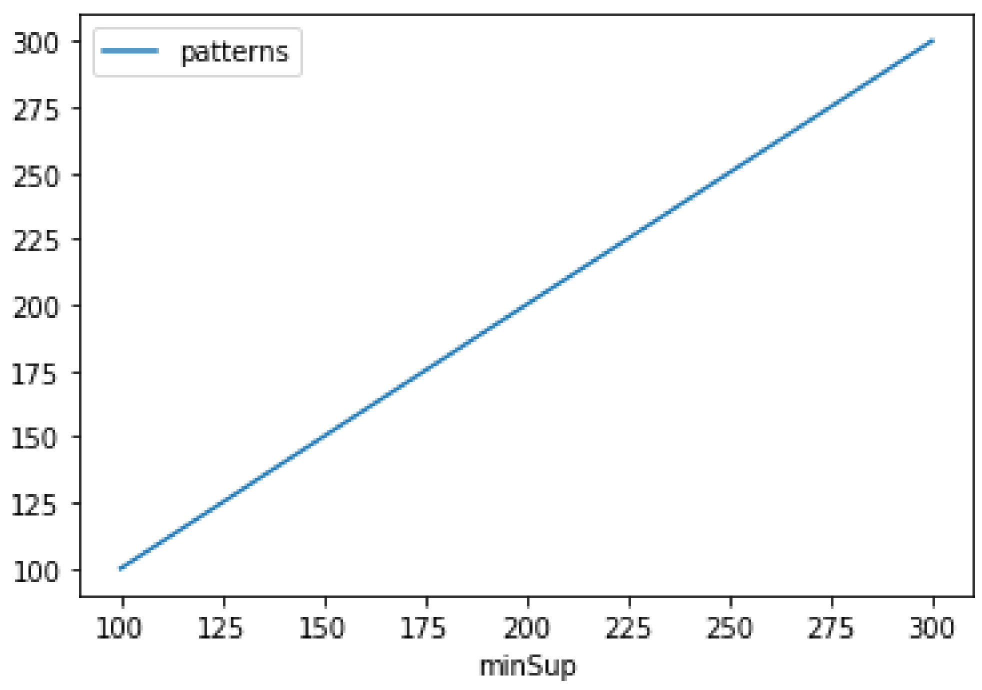
Step 5: Visualizing the results

Step 5.1 Importing the plot library

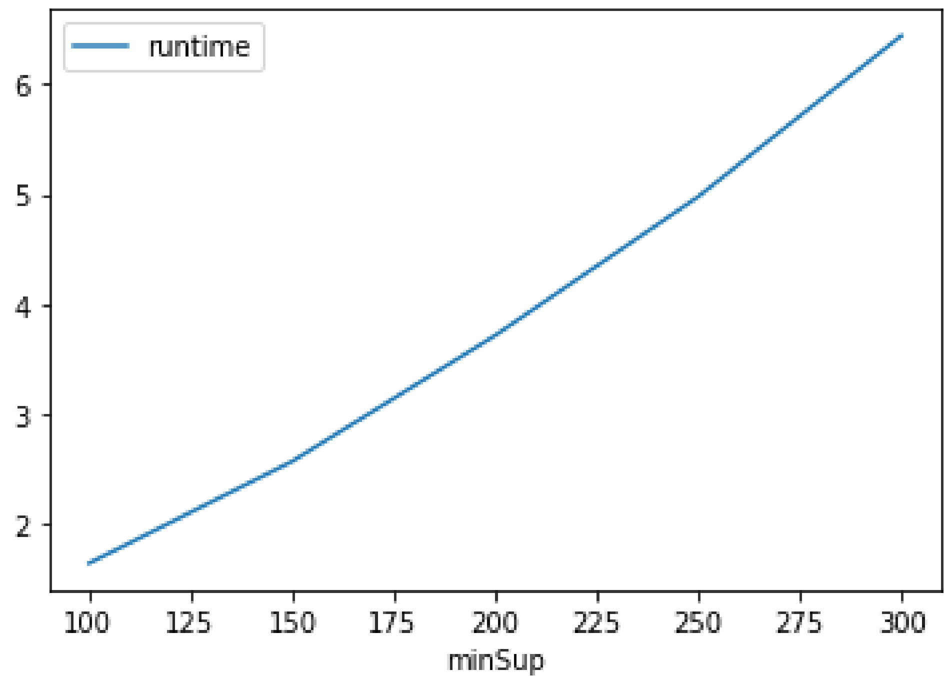
[5]:
```
from PAMI.extras.graph import plotLineGraphsFromDataFrame as plt
```

Step 5.2. Plotting the number of patterns

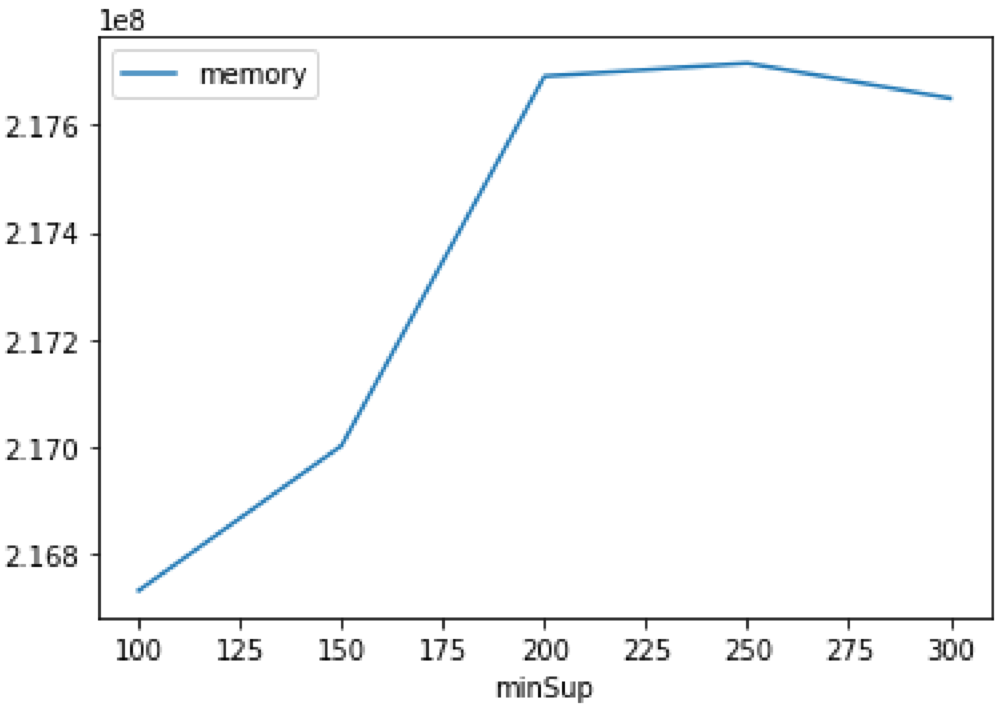
[6]:
```
ab = plt.plotGraphsFromDataFrame(result)
ab.plotGraphsFromDataFrame() #drawPlots()
```



```
Graph for No Of Patterns is successfully generated!
```



```
Graph for Runtime taken is successfully generated!
```



```
Graph for memory consumption is successfully generated!
```

1.0.1 Step 6: Saving the results as latex files

[7]:
```
from PAMI.extras.graph import generateLatexFileFromDataFrame as gdf
gdf.generateLatexCode(result)
```

```
Latex files generated successfully
```