Everett Hu
ehu6

CS 412 Assignment 3

Algorithm Descriptions:

Step 1.

I implemented the Apriori algorithm to mine frequent patterns. The basic steps are as follows:

- Retrieve list of all possible candidates along with their supports.
- Remove candidates that have a support less than the min_sup threshold to create a list of frequent itemsets.
- Add the patterns in the frequent itemset list to final list of frequent patterns.
- Create a new list of possible superset candidates based on the previous list of frequent patterns, making sure all possible subsets of each candidate exist in the previous list of frequent patterns.
- Remove candidates that have a support less than the min_sup threshold to create a new list of frequent itemsets.
- Add the patterns in the new list to the previous list of frequent patterns.
- Continue until no new patterns that meet the min_sup threshold can be generated.

Step 2.

To mine max patterns, I implemented a basic algorithm based on the output of Step1:

- Read in the frequent patterns from the output of step 1, partitioning the patterns based on the number of terms in each pattern
- Check all patterns for a superset that contains the current pattern and remove them from the list.
- Output all patterns that have no supersets that contain the pattern and are frequent.

To mine closed patterns, I implemented a similar algorithm:

- Read in the frequent patterns from the output of step 1, partitioning the patterns based on the number of terms in each pattern.
- Compare support levels of each pattern to supersets and look for a superset that contains the pattern and has the same support level.
- Output all patterns that have no supersets that contain the pattern and have the same support level.

Step 3.

To mine at least 10 association rules, I set lowerBoundMinSupport to be 0.004 (same as the min_sup level used for the first two steps) and then adjusted minMetric until 10 association rules were found.

Step 4.

To calculate purity, I implemented the formula described in the assignment description. I retrieved the f(t,p) values from the output of step 1 and got f(t',p) by either searching for the pattern frequency in the topic-i.txt files or, if available, from the corresponding output of step 1. Then I calculated the values for [(f(t,p) + f(t',p)) / | D(t,t') |] and used the max value in determining the purity value for each frequent pattern. The patterns are sorted in descending order first by purity and then by support for any potential ties.

Questions to Ponder:

A. How do you choose min_sup for this task? Explain how you choose the min_sup in your report. Any reasonable choice will be fine.

   I selected min_sup to be 0.004 * the number of lines in each topic. I came to this value as a result of trial and error, first trying 0.05, 0.01, and then 0.005. 0.004 seemed to be relatively optimal as it generated a fair amount of frequent patterns including those with up to three terms as well as taking a reasonable amount of time to run.

B. Can you figure out which topic corresponds to which domain based on patterns you mine? Write your observations in the report.

   Yes, the topic becomes obvious just looking at the first few entries in each pattern-i.txt file.

   topic-0.txt (data, mining, algorithm, graph) corresponds to Data Mining (DM)

   topic-1.txt (learning, using, module, based) corresponds to Machine Learning (ML)

   topic-2.txt (web, information, retrieval) corresponds to Information Retrieval (IR)

   topic-3.txt (database, system, knowledge, learning, data, logic) corresponds to Theory (TH)

   topic-4.txt (query, database, data, system, processing, distributed) corresponds to Database(DB)

C. Compare the result of frequent patterns, maximal patterns and closed patterns, is the result satisfying? Write down your analysis.

   Yes, the result is satisfying. The mined frequent patterns all match within each topic and clearly point to one of the domains. The maximal pattern mining results reduced the overall size of patterns a significant amount with respect to the frequent patterns. The closed frequent pattern mining results also reduced the overall size of patterns a significant amount, but less so than the maximal pattern mining, which was to be expected.

D. Topic 0 (lowerBoundMinSupport = 0.004, minMetric = 0.7):

```
 1. [data=1, series=1]: 48 ==> [time=1]: 48   <conf:(1)> lift:(17.94) lev:(0) conv:(45
 2. [series=1]: 209 ==> [time=1]: 194   <conf:(0.93)> lift:(16.65) lev:(0.02) conv:(12
 3. [lower=1]: 63 ==> [bound=1]: 57   <conf:(0.9)> lift:(49.95) lev:(0.01) conv:(8.84)
 4. [mining=1, sequential=1]: 47 ==> [pattern=1]: 42   <conf:(0.89)> lift:(17) lev:(0)
 5. [pattern=1, sequential=1]: 54 ==> [mining=1]: 42   <conf:(0.78)> lift:(6.72) lev:(
 6. [mining=1, rule=1]: 159 ==> [association=1]: 123   <conf:(0.77)> lift:(23.13) lev:
 7. [mining=1, association=1]: 159 ==> [rule=1]: 123   <conf:(0.77)> lift:(18.68) lev:
 8. [abstract=1]: 66 ==> [extended=1]: 49   <conf:(0.74)> lift:(106.56) lev:(0) conv:(
 9. [dimensionality=1]: 77 ==> [reduction=1]: 56   <conf:(0.73)> lift:(50.39) lev:(0.0
10. [itemsets=1]: 62 ==> [frequent=1]: 45   <conf:(0.73)> lift:(32.12) lev:(0) conv:(3
```

Topic 1 (lowerBoundMinSupport = 0.004, minMetric = 0.65):

```
 1. [machine=1, support=1]: 117 ==> [vector=1]: 115   <conf:(0.98)> lift:(42.26) lev:(
 2. [machine=1, vector=1]: 123 ==> [support=1]: 115   <conf:(0.93)> lift:(41.68) lev:(
 3. [markov=1, hidden=1]: 44 ==> [model=1]: 41   <conf:(0.93)> lift:(11.01) lev:(0) co
 4. [learning=1, semi=1]: 55 ==> [supervised=1]: 51   <conf:(0.93)> lift:(50.97) lev:(
 5. [model=1, hidden=1]: 46 ==> [markov=1]: 41   <conf:(0.89)> lift:(51.63) lev:(0) co
 6. [neighbor=1]: 137 ==> [nearest=1]: 121   <conf:(0.88)> lift:(60.6) lev:(0.01) conv
 7. [nearest=1]: 141 ==> [neighbor=1]: 121   <conf:(0.86)> lift:(60.6) lev:(0.01) conv
 8. [neural=1]: 128 ==> [network=1]: 101   <conf:(0.79)> lift:(16.49) lev:(0.01) conv:
 9. [vector=1, support=1]: 146 ==> [machine=1]: 115   <conf:(0.79)> lift:(24.66) lev:(
10. [support=1]: 217 ==> [vector=1]: 146   <conf:(0.67)> lift:(28.93) lev:(0.01) conv:
```

Topic 2 (lowerBoundMinSupport = 0.004, minMetric = 0.6):

```
 1. [answering=1]: 88 ==> [question=1]: 77   <conf:(0.88)> lift:(72.02) lev:(0.01) cor
 2. [page=1]: 131 ==> [web=1]: 107   <conf:(0.82)> lift:(6.63) lev:(0.01) conv:(4.59)
 3. [natural=1]: 228 ==> [language=1]: 170   <conf:(0.75)> lift:(15.15) lev:(0.02) cor
 4. [information=1, language=1]: 75 ==> [retrieval=1]: 55   <conf:(0.73)> lift:(6.56)
 5. [retrieval=1, content=1]: 58 ==> [based=1]: 42   <conf:(0.72)> lift:(8.36) lev:(0)
 6. [retrieval=1, language=1]: 77 ==> [information=1]: 55   <conf:(0.71)> lift:(5.87)
 7. [site=1]: 94 ==> [web=1]: 65   <conf:(0.69)> lift:(5.62) lev:(0.01) conv:(2.75)
 8. [engine=1]: 181 ==> [search=1]: 122   <conf:(0.67)> lift:(9.49) lev:(0.01) conv:(2
 9. [information=1, model=1]: 70 ==> [retrieval=1]: 46   <conf:(0.66)> lift:(5.87) lev
10. [question=1]: 121 ==> [answering=1]: 77   <conf:(0.64)> lift:(72.02) lev:(0.01) co
```

Topic 3 (lowerBoundMinSupport = 0.004, minMetric = 0.5):

```
1. [database=1, oriented=1]: 75 ==> [object=1]: 63   <conf:(0.84)> lift:(38.27) lev:(0
2. [satisfaction=1]: 96 ==> [constraint=1]: 80   <conf:(0.83)> lift:(19.88) lev:(0.01)
3. [artificial=1]: 79 ==> [intelligence=1]: 65   <conf:(0.82)> lift:(73.34) lev:(0.01)
4. [database=1, object=1]: 84 ==> [oriented=1]: 63   <conf:(0.75)> lift:(46.75) lev:(0
5. [expert=1]: 125 ==> [system=1]: 82   <conf:(0.66)> lift:(7.18) lev:(0.01) conv:(2.5
6. [oriented=1]: 163 ==> [object=1]: 102   <conf:(0.63)> lift:(28.51) lev:(0.01) conv:
7. [deductive=1]: 85 ==> [database=1]: 53   <conf:(0.62)> lift:(5.9) lev:(0) conv:(2.3
8. [object=1, oriented=1]: 102 ==> [database=1]: 63   <conf:(0.62)> lift:(5.84) lev:(0
9. [intelligence=1]: 114 ==> [artificial=1]: 65   <conf:(0.57)> lift:(73.34) lev:(0.01
```

Topic 4 (lowerBoundMinSupport = 0.004, minMetric = 0.7):

```
 1. [answering=1]: 64 ==> [query=1]: 62   <conf:(0.97)> lift:(5.57) lev:(0.01) conv:(1
 2. [database=1, oriented=1]: 96 ==> [object=1]: 93   <conf:(0.97)> lift:(18.06) lev:(
 3. [database=1, concurrency=1]: 49 ==> [control=1]: 45   <conf:(0.92)> lift:(23.67) l
 4. [materialized=1]: 56 ==> [view=1]: 51   <conf:(0.91)> lift:(33.21) lev:(0.01) conv
 5. [concurrency=1]: 133 ==> [control=1]: 107   <conf:(0.8)> lift:(20.73) lev:(0.01) c
 6. [oriented=1]: 183 ==> [object=1]: 141   <conf:(0.77)> lift:(14.37) lev:(0.01) conv
 7. [warehouse=1]: 73 ==> [data=1]: 56   <conf:(0.77)> lift:(7.26) lev:(0) conv:(3.63)
 8. [database=1, control=1]: 60 ==> [concurrency=1]: 45   <conf:(0.75)> lift:(55.52) l
 9. [expansion=1]: 84 ==> [query=1]: 60   <conf:(0.71)> lift:(4.11) lev:(0) conv:(2.78
10. [processing=1, efficient=1]: 70 ==> [query=1]: 50   <conf:(0.71)> lift:(4.11) lev:
```

Looking at the results, all of the support numbers are the same as those generated the pattern-i.txt files. In addition, all the association rules dealt with patterns mined as frequent patterns in step 1. The phrases also seem to make sense, for example the association rules between intelligence and artificial, warehouse and data, oriented and object, lower and bound, etc.

Source Files and Steps:

Step 1. apriori.py, takes parameters --dataFile, --outputFile, --minSup, generating frequent patterns for dataFile and writing those patterns sorted by support in descending order to the outputFile. Used this file to generate frequent patterns file pattern-i.txt for each topic in the patterns folder.

Step 2. max.py, closed.py mine maximal and closed patterns, respectively. max.py writes output files max-i.txt for each topic in the max folder. closed.py writes output files closed-i.txt for each topic in the closed folder. All output files are sorted by descending order of support.

Step 4. purity.py writes output files purity-i.txt into folder purity sorted first by descending order of purity, then support for ties.