Assignment 3 Report

Peng Li (peng24)

1. **Briefly explain what algorithms you use in Step4~Step6.**

   Step 4: I am using Apriori Algorithm to mine the frequent patterns for each topic. The Apriori Algorithm use bottom up approach, where frequent subsets are extended one item at a time. The minimum support is 0.1%. Each topic generates around 60 to 80 patterns that satisfies minimum support.

   Step 5: The program reads frequent pattern files that are generated in Step 4 and then filter out unqualified patterns. One interesting observation is that the size of the closed-pattern file doesn't reduce compared with frequent-pattern files. This is because the minimum support (1%) is relative big for this dataset. I re-ran step 4 and 5 with minsup = 0.1% and minsup = 0.01%. Both max-pattern and closed-pattern files are reduced a lot compared with the frequent pattern files.

   Step 6: purity(p,t)=log [f(t,p) / | D(t) |] - log (max [(f(t,p) + f(t',p)) / | D(t,t') |])
   Where f(t,p) / |D(t)| is the support value that is generated in Step 4. The only confusing part of this algorithm is D(t, t'). D(t, t') is the union of titles that include pattern p for topic t and t' (t ≠ t'). I recycled the code from Step 3, which read *word-assignments.dat* and save the title and an auto-generated unique id for each title in five HashTables according to five topics. Then I put the unique id into a HashSet if the corresponding title has pattern *p*. The size of the HashSet is |D(t, t')|.

   The order of patterns are sorted by Purity(normalized using 0-1 Normalization)*Support. I choose to order in this way because, the frequent pattern with higher purity will rank higher.

2. **Answer all the questions in Question to ponder.**
   Question to ponder A: How you choose min_sup for this task?

   I choose 1% as minimum support for this task. Using 1% generates around 70 to 80 frequent patterns. It is a reasonable number for us to find some useful information from the output. Additionally, I tried 0.1% and 0.01% as well, which generates 2,000 patterns and 9,000 patterns for each topic accordingly. These two minsup values are two small. Especially, when mining the completeness, it takes a huge amount of time to run.

   Question to ponder B: Can you figure out which topic corresponds to which domain based on patterns you mine?

Í0: Database
1: Information Retrieval
2: Machine Learning
3: Data Mining
4: Theory

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

Based on minsup = 1%, frequent pattern result is good. From the .phrase file we can get some useful information. Also, the lossy compression max-pattern is good. It reduced the size of frequent pattern. However, the lossess compression closed-pattern is the same as the frequent pattern. Thus, the algorithm doesn't reduce the size of the frequent pattern. So it is not satisfying. But when I changed minsup = 0.1% or minsup = 0.01%, the closed-pattern file size reduced a lot, which have good result.

3. **List your source file names and their corresponding Steps.**
   Step 2:
      Preprocessing.java
   Step 3:
      Partitioning.java
   Step 4:
      MiningFP.java
   Step 5:
      MiningMaxCloP.java
   Step 6:
      MiningPurityP.java
   Step 7:
      MiningPhrasenessP.java
      MiningCompletenessP.java
      CombinedRankingFunc.java
   Step Mapping Number to Terms:
      MapNumTerm.java

4. **Bonus:**
Based on KERT: Automatic Extraction and Ranking of Topical Keyphrases from Content-Representative Document Titles, I implemented the combined ranking.

$$r_t(p) = \begin{cases} 0 & \pi_t^{com} \le \gamma \\ \pi_t^{cov}[(1-\omega)\pi_t^{pur} + \omega\pi_t^{phr}](p) & \text{o.w.} \end{cases}$$

Where, $\gamma, \omega \in [0, 1]$ are two parameters. In the experiment, $\gamma = 0.5$, $\omega = 0.5$. γ shows how aggressively we prune the patterns based on completeness. This equation measures a weighted summation of two pointwise KL-divergence metrics.