Joseph Barbati
CS378

Apriori – Assignment 2

Algorithm:

- My algorithm follows the general algorithm. However, how I do each step might be of interest.
- Before I do anything with Apriori, I use a Data class I created to bring in the data from the file into a 2D list and call Apriori with that 2D list.
- Frequent 1-itemsets O(N*M) (N is num of rows, M is number of items in transaction):
  - This method takes the 2D list and visits each number and adds it as a key into a
  - dictionary (hashmap) where the value is the frequency of the itemset (or number in
  - this special case).
  - At each number I check if that number is in the dictionary (O(1)), and add 1 if it is. If
  - not, I add it to the dictionary with value initialized to 1.
  - After going through all the data, I call a method to "clean" the data.
    - This method simply goes through the data again and removes any number that is not in the frequent 1-itemset list, because I know if the 1-itemset is not frequent, any superset that includes this number cannot be frequent.
- Getting Candidate k-Itemsets - O(N*k) where N is the number of length k subsets in the row.
  - Originally, my plan was to look at each row and compute all length k subsets and insert them into a dictionary, similar to frequent 1-itemset function. This was horribly inefficient, so I modified it a little bit.
  - I recursively build the subset, however, during the recursive call, we check each time a number is added whether or not that tuple exists in Frequent n-itemsets dictionary (O(1)) where n is the current length of the subset I am building. If it exists in the frequent n-itemsets dictionary, I continue to build the subset, if not I return as I know any superset of the current subset will not be frequent. This modification increased efficiency exponentially.
  - Another efficiency consideration: Say my algorithm is currently looking for 5-itemsets, if we reach a transaction with only 4 items, we remove the row, as we will never need to look at that row again.
  - Immediately after getting the candidate k-itemsets for a row, I add it to a temporary dictionary and initialize the value to 1, unless it already exists in the dictionary in which case I increment the value by 1.
  - This temporary dictionary lasts the entire time we check for candidate k-itemsets. Once I go through all the data and get the frequencies of all candidate k-itemsets, I go through each key (itemset) in the temporary dictionary and if the value is greater than or equal to the minimum support, I put it into to the frequent k-itemset dictionary where the key is k and the value is a dictionary of key-value pairs where their keys are the itemsets and the values are the frequency. O(N) where N is the number of keys in the temporary dictionary. I then increment k and continue until the frequent k-itemset returned is empty.

- After Apriori is done, I send the results dictionary to an Output class along with the output filename and write the output data to the specified file.

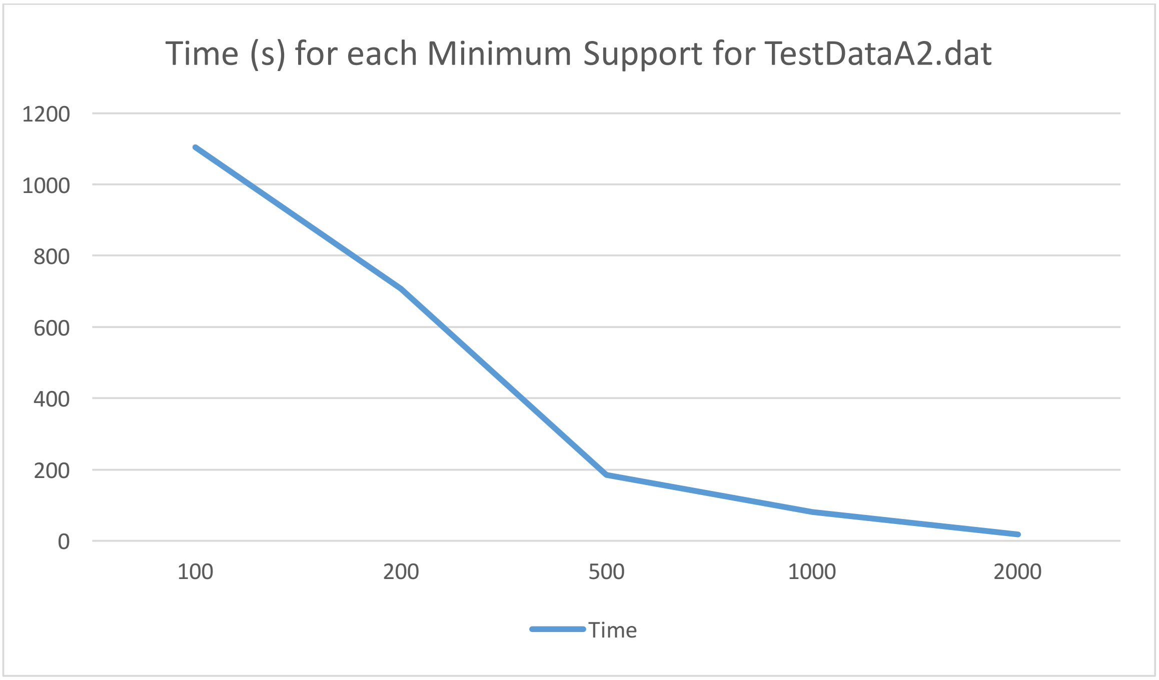

Figure 1. Times taken for a certain minimum support on a mid-2009 Macbook (2.26 GHz, 8GB RAM). Y-axis: Time (s), X-axis: Minimum Supoort.