University of Ioannina,
Computer Science Department

Data Mining – CS 059
Fall 2012

Assignment 1

This is the second part of Assignment 1. The deadline for this part is at the beginning of the class on November 6th. You should turn in the code for question 3, and either turn-in the remaining questions, or submit them on paper. For late submissions the late policy on the page of the course will be applied. The details for the turn-in are on the page of the course.

Question 1

A characteristic rule is a rule of the form $\{q\} \rightarrow \{p_1, p_2, \ldots, p_n\}$, where the left-hand-side of the rule has a single item. Given an itemset of k items, we can generate k such characteristic rules. We define the measure ξ of an itemset as the minimum confidence c of any of these characteristic rules. That is,

$$\xi(p_1, p_2, \ldots, p_k) = \min\{c(\{p_1\} \rightarrow \{p_2, \ldots, p_k\}), c(\{p_2\} \rightarrow \{p_1, \ldots, p_k\}), \ldots, c(\{p_k\} \rightarrow \{p_1, \ldots, p_{k-1}\})\}$$

Is the measure ξ monotone, anti-monotone, or non-monotone (neither monotone, nor anti-monotone)? Recall that for two sets X and Y, such that $X \subseteq Y$, a measure f is monotone if $f(X) \leq f(Y)$, and anti-monotone if $f(X) \geq f(Y)$.

Question 2

Consider the case where we have a dataset consisting of an *ordered sequence of events* (e.g., words in a document, letters in a string, or queries in a search log). We want to find all frequent *subsequences* (also called *serial episodes*) of events in the data that occur within a window *W* (that is, the difference in time between the first and the last event in the sequence is at most *W*). For example, <A,B,C> is a subsequence of length 3 that occurs within a window of size *W = 5*, that contains the sequence <A,A,B,D,C>. The subsequence actually occurs twice, once starting at the first position, and once starting at the second position. The support of a subsequence is the number of times that it is appears in the full sequence.

Describe a level-wise algorithm for computing all frequent subsequences with support at least *minsup*, for some given window size *W*. Describe clearly how you generate the candidate itemsets, and how you find the frequent itemsets. Your algorithm must be efficient in terms of memory and time, similar to the Apriori algorithm. You can describe your algorithm in pseudocode or in English.

Apply your algorithm on the sequence ABCACDBAAC with *W=4* and *minsup* = 2, and report the output of all intermediate steps of your algorithm.

Question 3

The goal of this question is to use frequent itemset mining in practice.

You have the following options for the input dataset:

1. A collection of 6500 Twitter user profiles collected by a research team in Korea. The details on how to obtain the data is on the web page of the course. In this dataset a "basket" is a Twitter user profile, and the "items" are words. The data needs to be preprocessed so that noisy data is removed, the words are normalized (no punctuation, lower case), and the English stop-words are removed. A list of stop words can also be found at the web page of the course.
2. Download your sent emails and find emails sent to at least 2 people. Create a dataset where each "basket" is one email, and the "items" are the email addresses. Your dataset should consist of at least 1000 "baskets", preferably more.
3. Suggest a dataset that you think is interesting to mine. The dataset should have at least 1000 "baskets", preferably more. Contact me with your suggestion.

Once you have decided on the dataset, you will use an existing software package for mining frequent itemsets. You can use either an implementation from FIMI repository, or the WEKA software. Details on how to download the software are on the page of the course. You will need to transform your data to the format required by each package.

Run the code for obtaining the frequent itemsets using a high enough support threshold that does not produce a huge amount of frequent itemsets. If the software you chose supports it, then output closed and maximal itemsets. Transform the output back to a readable form (e.g., transform items from ids to words). Then inspect the results and comment on itemsets that you found.

You should turn in the following:

- The preprocessed input file
- The produced itemsets
- A short report where you describe the input file, how you did the preprocessing, the choice of support threshold, and the commentary on the output.

Technical details

For pre-processing the data, some unix commands that you may find useful are the following:

a. cut: allows you to get specific columns from delimited data
b. sort: sorts the rows of a file in lexicographic order, –n for numeric
c. uniq: merges consecutive rows of a file that are identical.
d. grep: finds a sting within a file