Elliott King

CSE 469

Assignment 2 – Association Analysis

Frequent Itemsets:

L1:

{gene_1 }
{gene_3 }
{gene_4 }
{gene_5 }
{gene_6 }
{gene_8 }
{gene_9 }
{gene_12 }
{gene_14 }
{gene_17 }
{gene_21 }
{gene_22 }
{gene_23 }
{gene_25 }
{gene_26 }
{gene_27 }
{gene_31 }
{gene_36 }
{gene_37 }
{gene_39 }
{gene_43 }
{gene_45 }
{gene_47 }
{gene_48 }
{gene_50 }
{gene_53 }
{gene_54 }
{gene_55 }
{gene_56 }
{gene_59 }
{gene_60 }
{gene_63 }
{gene_64 }
{gene_66 }
{gene_67 }
{gene_71 }
{gene_72 }
{gene_75 }
{gene_77 }
{gene_78 }
{gene_81 }
{gene_83 }
{gene_84 }
{gene_87 }
{gene_89 }
{gene_90 }
{gene_91 }
{gene_93 }
{gene_94 }
{gene_98 }
{gene_99 }

L2:

{gene_1 , gene_3 }
{gene_1 , gene_5 }
{gene_1 , gene_6 }
{gene_1 , gene_8 }
{gene_1 , gene_21 }
{gene_1 , gene_47 }
{gene_1 , gene_54 }
{gene_1 , gene_59 }
{gene_1 , gene_67 }
{gene_1 , gene_72 }
{gene_1 , gene_81 }
{gene_1 , gene_84 }
{gene_1 , gene_87 }
{gene_1 , gene_89 }
{gene_1 , gene_91 }
{gene_1 , gene_94 }
{gene_3 , gene_5 }
{gene_3 , gene_47 }
{gene_3 , gene_59 }
{gene_3 , gene_72 }
{gene_5 , gene_6 }
{gene_5 , gene_47 }
{gene_5 , gene_59 }
{gene_5 , gene_72 }
{gene_5 , gene_87 }

{gene_5 , gene_91 }

{gene_6 , gene_59 }

{gene_59 , gene_72 }

{gene_59 , gene_87 }

L3:

{gene_1 , gene_3 , gene_5 }

{gene_1 , gene_59 , gene_72 }

Length-3 Candidate Itemsets:

C3:

{gene_1 , gene_3 , gene_5 }

{gene_1 , gene_3 , gene_47 }

{gene_1 , gene_3 , gene_59 }

{gene_1 , gene_3 , gene_72 }

{gene_1 , gene_5 , gene_6 }

{gene_1 , gene_5 , gene_47 }

{gene_1 , gene_5 , gene_59 }

{gene_1 , gene_5 , gene_72 }

{gene_1 , gene_5 , gene_87 }

{gene_1 , gene_5 , gene_91 }

{gene_1 , gene_6 , gene_59 }

{gene_1 , gene_59 , gene_72 }

{gene_1 , gene_59 , gene_87 }

{gene_3 , gene_5 , gene_47 }

{gene_3 , gene_5 , gene_59 }

{gene_3 , gene_5 , gene_72 }

{gene_3 , gene_59 , gene_72 }

{gene_5 , gene_6 , gene_59 }

{gene_5 , gene_59 , gene_72 }

{gene_5 , gene_59 , gene_87 }

Apriori Implementation (In Java):

```
private static void run_apriori(String chosen) throws IOException {
            ArrayList<Integer[]> Frequent_Itemsets = new ArrayList<Integer[]>();
            ArrayList<Integer[]> Candidate_Itemsets = new ArrayList<Integer[]>();

            // First, find frequent items at level one.
            // These are the bases for the apriori algorithm.
            // For each level, to be neat, we will write a new file.

            File fout = new File(chosen + "_frequents");
            FileOutputStream ffos = new FileOutputStream(fout);

            File cout = new File(chosen + "_candidates");
            FileOutputStream cfos = new FileOutputStream(cout);

            BufferedWriter frequentw = new BufferedWriter(new
OutputStreamWriter(ffos));
            BufferedWriter candidatew = new BufferedWriter(new
OutputStreamWriter(cfos));

            frequentw.write("L1: ");
            frequentw.newLine();
            // I could not figure out a way to increment level and run through the
appropriate
            // number of candidates simply using loops, so I manually coded this
for itemsets
            // of two, and itemsets of three. Definitely a big bummer, if I had
more time I
            // could figure it out. It would probably have to be recursive.

            for(int j = 0; j < columns; j++){
                  if(Support[j] >= minsup){
                        Frequent_Itemsets.add(new Integer[]{j});
                        frequentw.write("{" + Headers[j] + "}");
                        frequentw.newLine();
                  }
            }
            frequentw.newLine();

            // Candidate and frequent itemsets of level two:
            // Keep in mind that the candidate sets are never stored, only the
frequent
            // Candidate sets are, however, transcribed to the appropriate file

            candidatew.write("C2: ");
            candidatew.newLine();
            frequentw.write("L2: ");
            frequentw.newLine();

            for(int i = 0; i < Frequent_Itemsets.size(); i++){
                  int col1 = Frequent_Itemsets.get(i)[0];
                  for(int j = i + 1; j < Frequent_Itemsets.size(); j++){
                        int col2 = Frequent_Itemsets.get(j)[0];
                        candidatew.write("{" + Headers[col1] + ", " +
Headers[col2] + "}");
                        candidatew.newLine();
                        int support = 0;
                        for(int k = 0; k < Data.length; k++){
                              if(Data[k][col1] > 0 && Data[k][col2] > 0) support+
+;
                        }
                        if(support >= minsup){
                              Candidate_Itemsets.add(new Integer[]{col1,col2});
                              frequentw.write("{" + Headers[col1] + ", " +
Headers[col2] + "}");
                              frequentw.newLine();
                        }
                  }
            }
            candidatew.newLine();
            frequentw.newLine();
            Frequent_Itemsets = Candidate_Itemsets;
            Candidate_Itemsets = new ArrayList<Integer[]>();

            // Candidate and frequent itemsets of level three:
            // We must make sure that the candidates are correctly chosen

            candidatew.write("C3: ");
            candidatew.newLine();
            frequentw.write("L3: ");
```

```
        frequentw.newLine();

        for(int i = 0; i < Frequent_Itemsets.size(); i++){
            int col1 = Frequent_Itemsets.get(i)[0];
            int col2 = Frequent_Itemsets.get(i)[1];

            for(int j = i + 1; j < Frequent_Itemsets.size(); j++){
                int col3;
                if(Frequent_Itemsets.get(j)[0] == col1){
                    col3 = Frequent_Itemsets.get(j)[1];
                    for(int k = j + 1; k < Frequent_Itemsets.size(); k +
+){
                        if(Frequent_Itemsets.get(k)[0] == col2){
                            if(Frequent_Itemsets.get(k)[1]==col3){
                                candidatew.write("{" +
Headers[col1] + ", " + Headers[col2] + ", " + Headers[col3] + "}");
                                candidatew.newLine();
                                int support = 0;
                                for(int row = 0; row < Data.length;
row++){
                                    if(Data[row][col1] > 0 &&
Data[row][col2] > 0 && Data[row][col3] > 0) support++;
                                }
                                if(support >= minsup){
                                    Candidate_Itemsets.add(new
Integer[]{col1,col2,col3});
                                    frequentw.write("{" +
Headers[col1] + ", " + Headers[col2] + ", " + Headers[col3] + "}");
                                    frequentw.newLine();
                                }
                            }
                        }
                    }
                }

            }
        }

        candidatew.close();
        frequentw.close();
    }
```