BIL713 Data Mining – Assignment 1

Mehmet Emin Mumcu – N16221464

Objectives

Association rule mining is an approach to identify similarities between different data. It provides insight about relationships of similar objects. It is widely used in recommendation engines or market analysis.

Dataset description

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

All attributes are used, since there is no reason to eliminate any of them for this dataset. Only the file format is changed to ".arff" just to provide conformity with the library. Dataset can be found in URL:

https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records.

Rule mining process

For rule mining, Weka java API 3.8.0 is used. This library provides implementations of both Apriori and FP-Growth algorithms. The "party" attribute is used as classifying attribute since it is appropriate to identify the tendency of parties for different votes. You can find appropriate code Appendix A.

Resulting rules

Apriori

Minimum support: 0.45 (196 instances)

Minimum metric <confidence>: 0.9

Number of cycles performed: 10

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 5

Size of set of large itemsets L(3): 1

Best rules found:

1. adoption-of-the-budget-resolution=y physician-fee-freeze=n 219 ==> Class=democrat 219 conf:(1)

2. adoption-of-the-budget-resolution=y physician-fee-freeze=n aid-to-nicaraguan-contras=y 198 ==> Class=democrat 198 conf:(1)

3. physician-fee-freeze=n aid-to-nicaraguan-contras=y 211 ==> Class=democrat 210 conf:(1)

4. physician-fee-freeze=n education-spending=n 202 ==> Class=democrat 201 conf:(1)

5. physician-fee-freeze=n 247 ==> Class=democrat 245 conf:(0.99)

6. el-salvador-aid=n aid-to-nicaraguan-contras=y 204 ==> Class=democrat 197 conf:(0.97)

7. el-salvador-aid=n 208 ==> Class=democrat 200 conf:(0.96)

8. adoption-of-the-budget-resolution=y aid-to-nicaraguan-contras=y 215 ==> Class=democrat 203 conf:(0.94)

9. education-spending=n 233 ==> Class=democrat 213 conf:(0.91)

10. adoption-of-the-budget-resolution=y 253 ==> Class=democrat 231 conf:(0.91)

FPGrowth

found 12 rules (displaying top 10)

Showing only rules that contain: Class

1. [el-salvador-aid=y, Class=republican]: 157 ==> [physician-fee-freeze=y]: 156 <conf:(0.99)> lift:(2.44) lev:(0.21) conv:(46.56)

2. [crime=y, Class=republican]: 158 ==> [physician-fee-freeze=y]: 155 <conf:(0.98)> lift:(2.41) lev:(0.21) conv:(23.43)

3. [Class=republican]: 168 ==> [physician-fee-freeze=y]: 163 <conf:(0.97)> lift:(2.38) lev:(0.22) conv:(16.61)

4. [physician-fee-freeze=y, Class=republican]: 163 ==> [el-salvador-aid=y]: 156 <conf:(0.96)> lift:(1.96) lev:(0.18) conv:(10.45)

5. [physician-fee-freeze=y, Class=republican]: 163 ==> [crime=y]: 155 <conf:(0.95)> lift:(1.67) lev:(0.14) conv:(7.79)

6. [Class=republican]: 168 ==> [crime=y]: 158 <conf:(0.94)> lift:(1.65) lev:(0.14) conv:(6.57)

7. [Class=republican]: 168 ==> [el-salvador-aid=y]: 157 <conf:(0.93)> lift:(1.92) lev:(0.17) conv:(7.18)

8. [el-salvador-aid=y, physician-fee-freeze=y]: 168 ==> [Class=republican]: 156 <conf:(0.93)> lift:(2.4) lev:(0.21) conv:(7.93)

9. [Class=republican]: 168 ==> [el-salvador-aid=y, physician-fee-freeze=y]: 156 <conf:(0.93)> lift:(2.4) lev:(0.21) conv:(7.93)

10. [crime=y, physician-fee-freeze=y]: 168 ==> [Class=republican]: 155 <conf:(0.92)> lift:(2.39) lev:(0.21) conv:(7.37)

Conclusion

The approach of two algorithms are different, so their results are different. FP-Growth seems much faster for this dataset. Since Apriori takes 188 milliseconds and FP-Growth takes 61 milliseconds. FP-Growth found 12 rules, where Apriori found 10 rules with confidence above 0.9.

Appendix A – Code

```
package memin.hacettepe.datamining.homeworks.hw1;

import weka.associations.Apriori;
import weka.associations.FPGrowth;
import weka.core.Instances;
import weka.core.converters.ConverterUtils;

import java.io.InputStream;

/**
 * Data mining Assignment-1.
 * Created by Memin on 10.04.2017.
 */
public class VoteAssociator {

    public static void main(String[] args) {
        InputStream stream =
VoteAssociator.class.getClassLoader().getResourceAsStream("vote.arff");

        // Read file
        Instances instances = null;
        try {
            ConverterUtils.DataSource source = new
ConverterUtils.DataSource(stream);
            instances = source.getDataSet();
        } catch (Exception e) {
            e.printStackTrace();
        }

        // find rules for Apriori Algorithm
        Apriori apriori = new Apriori();
        try {
            long startTime = System.nanoTime();
            apriori.mineCARs(instances);
            long endTime = System.nanoTime();
            long duration = (endTime - startTime);
            System.out.println("Apriori: " + duration / 1000000 + " ms");
            System.out.println(apriori.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }

        // find rules for FPGrowth Algorithm
        FPGrowth fpGrowth = new FPGrowth();
        fpGrowth.setRulesMustContain("Class");
        try {
            long startTime = System.nanoTime();
            fpGrowth.buildAssociations(instances);
            long endTime = System.nanoTime();

            long duration = (endTime - startTime);
            System.out.println("FPGrowth: " + duration / 1000000 + " ms");
            System.out.println(fpGrowth.toString());
        } catch (Exception e) {
            e.printStackTrace();
        }


    }

}
```

https://github.com/Memn/hacettepe-data-mining-hw1 contains the whole project.