Malware Classification

DIG DATA LABS

Manu Singh, Marshal Patel, Tejas Shah

May 8, 2015

1. Company Overview

1.1. Description

Dig Data Labs is a Bloomington based data analytics and consulting firm specializing in data mining and Big data consulting. We provide our services to various clients ranging from software security firms to pharmaceutical companies. We at Dig Data Labs aim at providing best data driven solutions at minimum cost

Members Currently the team of Dig Data Labs comprises of three passionate data scientists.

- **Manu Singh** : With over four years of industry experience, He is the co-founder and CEO of the company. His areas of expertise are Data mining, Software security and Image processing.
- **Marshal Patel**: Marshal comes with an industry experience of over two years, he is the Chief technology Officer here at Dig Data Labs. He is adept in using analytical tools like R and has a strong background in computer networks and database systems.
- **Tejas Shah**: Tejas recently graduated from Mumbai University and demonstrates stellar academic credentials in Computer Science and Engineering. He is an expert in algorithm design, artificial intelligence and distributed systems.

2. Client Overview

The project is contracted by Acme Dynamic General Systems(ADGS). ADGS is a leading IT solutions provider in the Mid-West area which builds software for zoos that run on Microsoft Windows operating system. They currently have over 20 products pertaining to zoo management.

ADGS currently aims at increasing the robustness of the their flagship product ZooKeeper against malicious attack by incorporating malware detection algorithm.

3. Executive Summary

Due to easy availability of open-source malware creation toolkits, it takes minimum efforts for a novice programmer to create and propagate malwares of any kind. Due to *polymorphic* and *metamorphic* nature of malware, detecting such malwares is a challenging task in terms of storage and computation which is not just limited to signature matching. In this report, we highlight our approach on how we attacked the problem of malware classification using data-mining techniques. Our approach helps us to accurately identify and classify malware with an impressive log-loss of **0.0625**(Please see Outcome section for more details).

The report is organized as follows: Section 4 consists of the past and present problems, Section 5 consists of the past and our approach to classify malware, Section 6 consists of the comparison of present and past results and outlines the outcome of our method.

4. Problem Definition

Commercial anti-malware systems were successful to a large extent in identifying static signature based malware. However, in the past decade, malware writers have developed novel ways of writing diverse variety of malwares which makes it difficult for the existing anti-malware systems to defend against emerging malwares. This challenge can be attributed to:

1. Code obfuscation: This consists of putting garbage code and to rearrange certain sections in the code to change it's signature making it untraceable.
2. Malwares detection has evolved from static signature based to dynamic behavior based.
3. Even if we manage to capture all the behavior patterns in the form of byte codes or assembly language, it is not feasible to perform classification over commodity hardware due to compute and storage limitations. This was the major challenge in this project due to large data size (training and testing).

   In order to effectively classify the malware our objective was twofold:

   a) Fit the input data with least error.

   b) Predict the class of unseen data.

5. Overview of the dataset

We were provided with a set of known malware files belonging to 9 different classes. Each malware file was identified by a an ID and a class (an integer value from 1 to 9). For each file we had a bytecode file and an assembly file, which was generated from IDA disassembler tool. Fig 5.1 shows the frequencies of files belonging to each class.

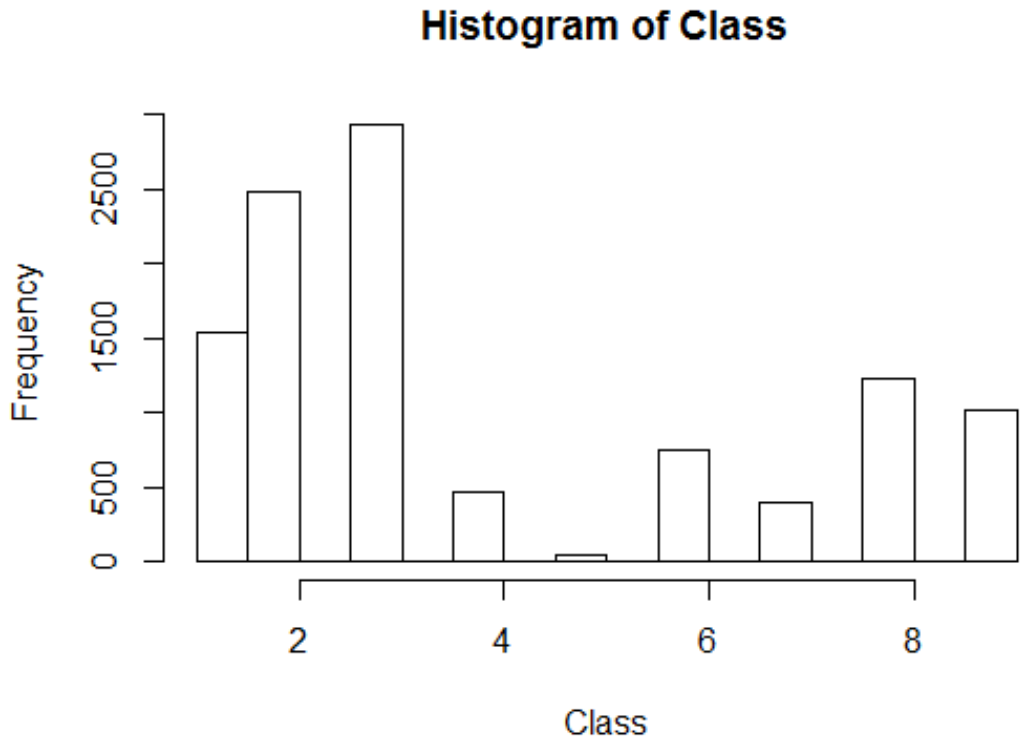

Figure 5.1: Histogram of Class

6. Solution

6.1. Past Approaches

1. ***Signature Based (Static Malware):*** Static malwares are identified by their unique signatures which makes it easier for anti-malware softwares to detect them. Even though this approach is quick and accurate yet, it has following limitations:
   a) The signature definitions of the anti-malware system must be up to date in order to defend against newly created malwares.
   b) This approach fails against polymorphic and metamorphic class of malwares which are uniquely identified by behavior rather than signatures.
2. ***Behavioral based (Dynamic Malware):*** This class of malware are detected using behavioral patterns of the malware. Hence, these malwares require a simulator to study the behavior of a suspected files.

6.2. Our Approach

We propose a unique and feasible approach which analyses the bytecode and assembly files and is computation and storage efficient. The key step in the entire process was feature extraction and selection.

The underlying assumption for all of our approaches was:

The frequency of bytecodes and opcodes in a particular file belonging to a class will be similar to the bytecode and opcode frequency in another file belonging to same class.

6.2.1. 4-grams of Bytecodes

In this approach, we first segregated the bytecode files on the basis of their classes and then extracted 4-grams from each file. Later, we attempted to find the intersection of all the ngram files of the same class in order to extract unique features for that particular class.

Steps:

a) Clustered the data based on the classification excel sheet-create nine folders and after preprocessing each byte file move them to respective folders.

b) Ran N-gram individually in each folder for all files and generated unique set of features and stored the results.

6.2.2. Bytecode frequency

This approach was also baselined on our key assumption. However, this time we only considered the byte frequency of all the files and created a frequency vector of each file and stored it into a single matrix. This matrix is than fed into the random forest classifier for the purpose of training. The same process was followed in testing datasets.

Steps:

a) Cleaned all the byte files by removing the addresses on each line

b) Then each line was split on the bases of space. Thus we can count the bytes that occur in each line

c) Once the file was processed entirely, the file name, the class, and the corresponding frequency of each byte was written in the matrix. Fig 6.1 shows the flow diagram of the steps.

Hence the final matrix consists of 257 features(256 byte features and 1 '?' feature) for all the files in the training data.

6.2.3. Bytecode with Opcode frequency

Feature Extraction: In this approach along with the byte frequency we also considered opcode frequency. Similar to the byte frequency, all the opcodes were

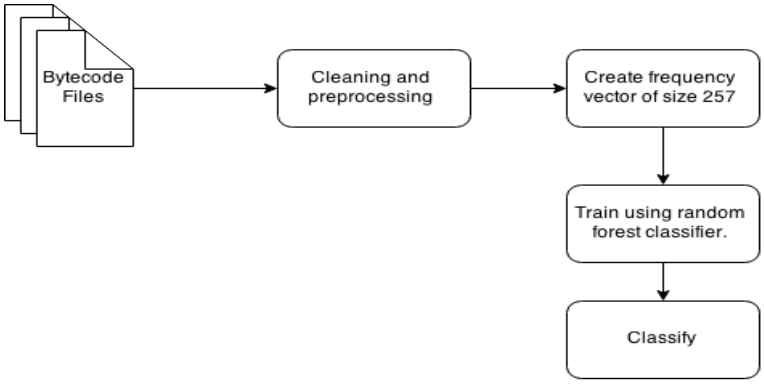

Figure 6.1: Steps: bytecode frequency

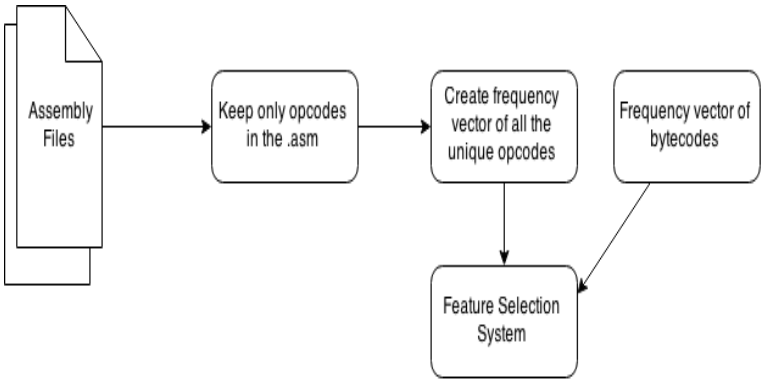

Figure 6.2: Steps: bytecode with opcode frequency: Feature Extraction

extracted from the corresponding assembly file. This extraction requires preprocessing of the assembly files. Later the byte frequencies were combined with the opcode frequency to form a training matrix.

Feature Selection: Once the features were extracted, to select the relevant features, we obtained the **information gain** of each feature. The features were sorted according to the increasing order of information gain. We selected the top 186 features from the sorted features. We further reduced the number of features on the basis of **correlation**. The threshold selected was 0.9. Consequently, the final feature count was 178.

Steps:

a) Byte frequencies were obtained.

b) Opcode features were obtained and merged with byte frequencies.

c) Using R's FSelector package, we obtained the information gain of all the features. We selected the top 186 features.

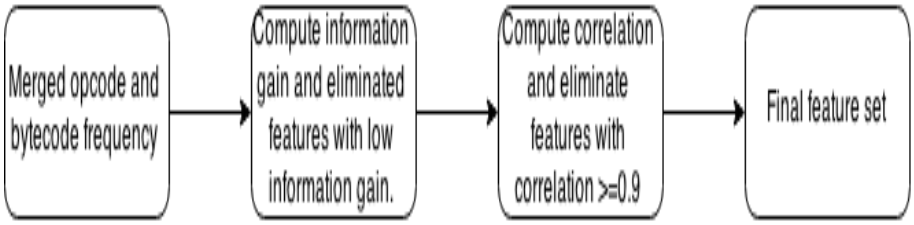

Figure 6.3: Steps: bytecode with opcode frequency: Feature Selection

d) Using R we eliminated the features which have high correlation between them.

e) The resulting features were used to train random forest classifier.

6.3. Random Forest Algorithm

We used random forest for classification because it reduces overfit and with sufficient number of trees generalization error is also reduced. We took around 150 trees in our classifier. It is based on ensemble model(combines the results from different models) that uses many decision tree based on feature bagging and randomly selecting features. Below algorithm-

a) Draw bootstrap samples from the original data.

b) For each of the bootstrap sample, train an unpruned classification tree.

   i. At each node, rather than choosing the best split among all predictors,choose a random subset of the features.

   ii. Choose the best split from among those sampled predictors variables.

c) Predict new data by aggregating the predictions of the tree.

7. Outcome

7.1. 4-grams of bytecode

This approach yielded null sets of intersection i.e. no common 4-grams for 66% of classes. As a result, we discontinued this approach and went ahead with an alternative. All the operations were automated by Python scripts which took around 9 hours to run on single machine.

7.2. Bytecode frequency

The bytecode files were processed using multi-threaded Java program. Classification was carried out using random forest function of scikit-learn package of python.

The train data was fed to random forest ensemble method, where in we achieved a cross validation error rate of **2.08%**

The test data was classified using the model generated. We were able to report a log-loss of **0.14** Below is the screen-shot of our submission:

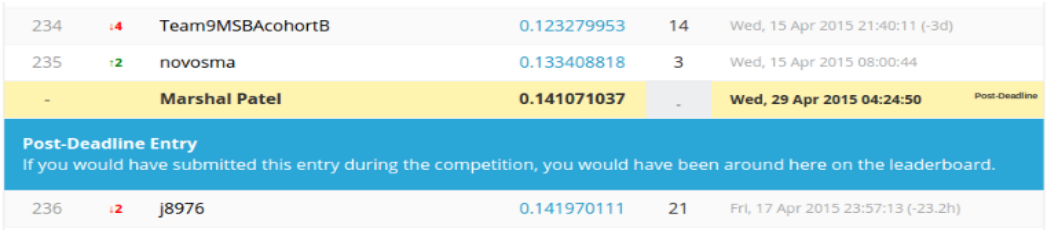

Figure 7.1: Rank-Bytecode frequency submission

7.3. Bytecode with Opcode frequency

The bytecode and assembly files were processed using multi-threaded Java program. Classification was carried out using random forest function scikit-learn package of python.

We achieved an improvement of over 50% in the log-loss with this approach and reported two results with a log-loss of **0.072** and **0.0625**.

The cross validation error was **1.03%**.

Figure 7.2: Rank-Bytecode with opcode frequency with 186 features

173	—	Tong	0.060056908	5	Sat, 11 Apr 2015 23:51:33 (-0.8h)
174	↑20	ML10	0.060783251	3	Wed, 11 Mar 2015 00:01:34
175	↑14	Nemesis	0.061088519	27	Thu, 02 Apr 2015 09:25:26
176	↓2	pku_cil	0.061626679	2	Fri, 06 Mar 2015 01:48:50
-		**Tejas**	**0.062529678**	-	**Mon, 04 May 2015 20:47:03** Post-Deadline
Post-Deadline Entry If you would have submitted this entry during the competition, you would have been around here on the leaderboard.					
177	↓170	Etor	0.063262432	190	Fri, 17 Apr 2015 11:26:23 (-3d)
178	↓8	Daniel Boline	0.063961135	2	Wed, 01 Apr 2015 12:29:09 (-11.9h)
179	↑1	ajitkumar.pu	0.065048160	21	Thu, 16 Apr 2015 09:16:49 (-24.7d)
180	↓4	iblackhurst	0.065772519	5	Thu, 16 Apr 2015 10:39:42 (-0.2h)

Figure 7.3: Rank-Bytecode with opcode frequency with 178 features

8. Future Work

Even though the results which we obtained were very impressive, we aim at further improving the log-loss by adapting the following approaches:

1. We aim to extract 4-grams of opcodes and apply feature engineering to combine extracted 4-grams with bytecode and opcode frequencies.

2. Another set of features which we wish to extract are bi-grams frequency of the bytecode files.

3. The signatures of the given malwares can be extracted and merged with the existing feature set to fine-tune the log-loss.

References

[1] http://101.datascience.community/tag/random-forest/.

[2] http://scikit-learn.org/stable/modules/ensemble.html.

[3] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.randomforestclassifier.htmll.

[4] Abdurrahman Pektaş, Mehmet Eriş, and Tankut Acarman. Proposal of n-gram based algorithm for malware classification. In *SECURWARE 2011, The Fifth International Conference on Emerging Security Information, Systems and Technologies*, pages 14–18, 2011.

[5] Babak Bashari Rad and Maslin Masrom. Metamorphic virus variants classification using opcode frequency histogram. *arXiv preprint arXiv:1104.3228*, 2011.

A. Source code for cleaning and pre-processing

ByteCleaningWorker.java

```
package bytes.frequency;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;


public class ByteCleaningWorker implements Runnable{

   private String filename;
   private String filePath;
   private BufferedReader br;
   private PrintWriter p;
   private static int threadcount = 0;
   private int threadName;
   private int mclass = 0;

   public ByteCleaningWorker(String filename, int mclass, PrintWriter p){
      this.filePath = "/home/vbox/test/"+filename;
      threadcount++;
      threadName = threadcount;
      this.mclass = mclass;
      this.p = p;
      this.filename = filename;
   }


   public void run() {
      System.out.println(threadName + "start");
      int x[] = new int[257];
      File input = new File(filePath);
      try {
         br = new BufferedReader(new FileReader(input));

      } catch (FileNotFoundException e) {
         e.printStackTrace();
      }

      String s = "";
      try {
```

```
            while((s = br.readLine())!=null){

               s = s.substring(9);
               String pk[] = s.split(" ");
               s = "";
               int j = 0;

               for(int i = 0; i<pk.length; i++){
                  if(pk[i].equals("??")){
                     x[256]++;
                  }
                  else{
                     j = Integer.parseInt(pk[i], 16);
                     x[j]++;
                  }
               }

            }
         } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
         }

         String out = filename + ",";
         for(int i = 0; i<x.length;i++){

            out+=x[i] + ",";
         }


         synchronized (p) {
            p.append(out + " \n");
         }

         System.out.println(threadName + "end");

         try {
            br.close();
         } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }


}
```

CleaningByte.java

```
package bytes.frequency;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import opcode.frequency.OpcodeFrequencyWorker;


public class CleaningByte {

   //private static PrintWriter p;
   private static HashMap<String, Integer> opcodes;

   public static void main(String args[]) throws FileNotFoundException {
      File f = new File("/home/vbox/test_asm.txt");
      Scanner sc = new Scanner(f);
      ExecutorService executor = Executors.newFixedThreadPool(1);
      int count = 0;
      try{
      // p = new PrintWriter(new FileOutputStream(new
          File("/home/vbox/outputOpcodes.csv"), true));
      }
      catch(Exception e){

      }
      createOpcodeMap();
      String head = "FileName,";
      Iterator it = opcodes.entrySet().iterator();
       while (it.hasNext()) {
         Map.Entry pair = (Map.Entry)it.next();
         head += pair.getKey() + ",";
       }
       p.write(head +"\n");
      String head = "FileName,";
      for(int i=0; i<257; i++){
         head+=i+",";
      }
      p.write(head +"\n");
```

```
   while (sc.hasNext() && count<1) {
      String string = sc.nextLine();
      //String x[] = string.split("\t");
      //if(x[1].equals("9")){
      Runnable worker = new OpcodeFrequencyWorker(string,
            Integer.parseInt(x[1]) p);
      executor.execute(worker);
      count++;
      //}
      //if(x[1].equals("8"))
         //break;

   }

   sc.close();
   executor.shutdown();
   while (!executor.isTerminated()) {
   }
   //p.close();

}
public static void createOpcodeMap(){
   opcodes = new HashMap<String, Integer>();
   opcodes.put("add", 0);
   opcodes.put("and", 0);
   opcodes.put("bound", 0);
   opcodes.put("bytes", 0);
   opcodes.put("call", 0);
   opcodes.put("cmp", 0);
   opcodes.put("dec", 0);
   opcodes.put("div", 0);
   //opcodes.put("dword", 0);
   //opcodes.put("db", 0);
   //opcodes.put("dw", 0);
   opcodes.put("extrn", 0);
   opcodes.put("idiv", 0);
   opcodes.put("imul", 0);
   opcodes.put("inc", 0);
   opcodes.put("iret", 0);
   opcodes.put("jcc", 0);
   opcodes.put("jmp", 0);
   opcodes.put("lea", 0);
   opcodes.put("leave", 0);
   opcodes.put("lock", 0);
   opcodes.put("ltr", 0);
   opcodes.put("mov", 0);
   opcodes.put("movs", 0);
   opcodes.put("movsx", 0);
   opcodes.put("movzx", 0);
```

```
        opcodes.put("mul", 0);
        opcodes.put("neg", 0);
        opcodes.put("nop", 0);
        opcodes.put("not", 0);
        opcodes.put("or", 0);
        opcodes.put("pop", 0);
        opcodes.put("push", 0);
        opcodes.put("ret", 0);
        opcodes.put("sub", 0);
        opcodes.put("test", 0);
        opcodes.put("xor", 0);
        opcodes.put("xchg", 0);
        //opcodes.put("dd", 0);
        opcodes.put("loc", 0);
        //opcodes.put("dll", 0);
        opcodes.put("jnb", 0);
        opcodes.put("jz", 0);
        opcodes.put("jnz", 0);
        opcodes.put("ja", 0);
        opcodes.put("jb", 0);
        opcodes.put("jl", 0);
        opcodes.put("js", 0);
        opcodes.put("shl", 0);
        opcodes.put("shr", 0);
    }

}
```

ASMCleaningWorker.java

```
package cleaning.asm;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class AsmCleaningWorker implements Runnable {
    private String filePath;
    private String outputfilepath;
    private Scanner sc;
    private PrintWriter p;
    private static int threadcount = 0;
    private int threadName;
    private BufferedReader br;
```

```
public AsmCleaningWorker(String filename) {
   this.filePath = "/home/vbox/test/" + filename;
   this.outputfilepath = "/home/vbox/" + filename;
   threadcount++;
   threadName = threadcount;
}

public void run() {
   System.out.println(threadName + "start");
   File f = new File(filePath);

   try {
      br = new BufferedReader(new FileReader(f));
      p = new PrintWriter(outputfilepath);
   } catch (FileNotFoundException e) {
      e.printStackTrace();
   }

   String s = "";
   try {

      while ((s = br.readLine()) != null) {

         int start = s.indexOf("\t\t");
         if (start != -1) {
            String n = s.substring(start);

            if (!(n.contains("; --") || n.contains("; =="))) {

               int end = n.indexOf(";");
               if (end != -1) {
                  if (!n.substring(0, end).trim().equals("")) {
                     p.write(n.substring(0, end).trim() + "\n");
                  }
               } else {
                  p.write(n.trim() + "\n");
               }
            } else {
               p.write(n.trim() + "\n");
            }
         }
      }
   } catch (IOException e) {
      e.printStackTrace();
   }

   try {
      br.close();
```

```
      } catch (IOException e) {
         e.printStackTrace();
      }
      p.close();
      System.out.println(threadName + "end");

   }

}
```

CleaningASM.java

```
package cleaning.asm;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import opcode.ngram.NgramFreqWorker;

public class CleaningAsm {

   public static void main(String args[]) throws FileNotFoundException,
         InterruptedException, ExecutionException {

      File f = new File("/home/vbox/test_asm.txt");
      Scanner sc = new Scanner(f);
      ExecutorService executor = Executors.newFixedThreadPool(500);
      int count = 0;

      while (sc.hasNext()) {
         String string = sc.nextLine();
         int  i = string.indexOf(".");
         //System.out.println(i);
         string = string.substring(0, i);

         //System.out.println(string);
      // String x[] = string.split("\t");
         Runnable worker = null;
      // if(x[1].equals("9")){
            worker = new NgramFreqWorker(string);
            executor.execute(worker);
            count++;
         //}
         //if(x[1].equals("9"))
            //break;
```

```
//      count++;

      }

      sc.close();
      executor.shutdown();
      while (!executor.isTerminated()) {
      }
   }

}
```

NgramFreqWorker.java

```
package opcode.ngram;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;

public class NgramFreqWorker implements Runnable{

   private PrintWriter ngramPrint;
   private static int threadcount;
   private int threadName;
   private int mclass;
   private BufferedReader br;
   private String filename;
   private HashMap<String, Integer> ngramFreq;

   public NgramFreqWorker(String filename) throws FileNotFoundException{

      ngramFreq = new HashMap<String, Integer>();
      threadcount++;
      threadName = threadcount;
      this.filename = filename;
   }
   @Override
   public void run() {

      File input = new File("/home/vbox/ngram/"+filename+".asm");
```

```
try {
   br = new BufferedReader(new FileReader(input));
   File n = new File("/home/vbox/ngramFreq/"+filename+".txt");
   ngramPrint = new PrintWriter(n);
} catch (FileNotFoundException e1) {
   e1.printStackTrace();
}

System.out.println(threadName + " start");
String line;

try {
      line = br.readLine();
   String s[] = line.split(" ");
   for (int i = 0; i < s.length - 4; i++) {
      String key = "";
      for (int j = i; j < i + 4; j++) {
         key += s[j] + " ";

      }
      if (!ngramFreq.containsKey(key)) {
         ngramFreq.put(key, 1);
      } else {
         int x = ngramFreq.get(key);
         ngramFreq.put(key, ++x);
      }
   }

   String out = "";
   Iterator it = ngramFreq.entrySet().iterator();
   int ss = 0;
   while (it.hasNext()) {
      Map.Entry pair = (Map.Entry) it.next();
      //if ((Integer) pair.getValue()>1) {
         out += pair.getKey() + ":" + pair.getValue() + "\n";
         ss++;
      //}
   }
   ngramPrint.write(out);

}catch (Exception e) {
   System.out.println(this.filename);
   e.printStackTrace();

}
finally{
   ngramPrint.close();
}
System.out.println(threadName+" end");
```

```
    }

}
```

OpcodeFrequencyWorker.java

```
package opcode.frequency;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;

public class OpcodeFrequencyWorker implements Runnable{

    //private PrintWriter p;
    private PrintWriter ngramPrint;
    private static int threadcount;
    private int threadName;
    private int mclass;
    private BufferedReader br;
    private HashMap<String, Integer> opcodes;
    //private HashSet<String> missed = new HashSet<String>();
    private String filename;


    public OpcodeFrequencyWorker(String filename, int mclass/*, PrintWriter
        p*/) throws FileNotFoundException{

        File n = new File("/home/vbox/ngram/"+filename);
        ngramPrint = new PrintWriter(n);
        threadcount++;
        threadName = threadcount;
        this.mclass = mclass;
        //this.p = p;
        this.filename = filename;
        createOpcodeMap();
    }


    @Override
    public void run() {
        File input = new File("/home/vbox/CleanedASM/"+filename+".asm");
```

```
try {
   br = new BufferedReader(new FileReader(input));
} catch (FileNotFoundException e1) {
   e1.printStackTrace();
}
System.out.println(threadName + " start");
String newLine ="";
String out = filename+ ",";
try {
   while((newLine=br.readLine())!= null){
      String s[] = newLine.split(" ");

      if(opcodes.containsKey(s[0])){
         ngramPrint.write(s[0] + " ");
         int i = opcodes.get(s[0]);
         opcodes.put(s[0], ++i);
      }
      else if(s.length>2){
         if(opcodes.containsKey(s[1])){
            ngramPrint.write(s[1] + " ");
            int i = opcodes.get(s[1]);
            opcodes.put(s[1], ++i);
         }
         if(opcodes.containsKey(s[2])){
            ngramPrint.write(s[2] + " ");
            int i = opcodes.get(s[2]);
            opcodes.put(s[2], ++i);
         }
      }
      /*else{
         missed.add(s[0]);
      }*/
   }
} catch (IOException e) {
   e.printStackTrace();
}
finally{
   try {
      br.close();
   } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   }

   ngramPrint.close();
}

/*Iterator it = opcodes.entrySet().iterator();
 while (it.hasNext()) {
```

```
      Map.Entry pair = (Map.Entry)it.next();
      out += pair.getValue()+ ",";
    }*/
    /*Iterator<String> itr2 = missed.iterator();*/
     /*while(itr2.hasNext()){
        System.out.print(itr2.next() + "\n");
     }
     System.out.println();
     */
    /* synchronized (p) {
      p.append(out + "\n");

   }*/


   System.out.println(threadName+" end");
}

public void createOpcodeMap(){
   opcodes = new HashMap<String, Integer>();
   opcodes.put("add", 0);
   opcodes.put("and", 0);
   opcodes.put("bound", 0);
   opcodes.put("bytes", 0);
   opcodes.put("call", 0);
   opcodes.put("cmp", 0);
   opcodes.put("dec", 0);
   opcodes.put("div", 0);
   //opcodes.put("dword", 0);
   //opcodes.put("db", 0);
   //opcodes.put("dw", 0);
   opcodes.put("extrn", 0);
   opcodes.put("idiv", 0);
   opcodes.put("imul", 0);
   opcodes.put("inc", 0);
   opcodes.put("iret", 0);
   opcodes.put("jcc", 0);
   opcodes.put("jmp", 0);
   opcodes.put("lea", 0);
   opcodes.put("leave", 0);
   opcodes.put("lock", 0);
   opcodes.put("ltr", 0);
   opcodes.put("mov", 0);
   opcodes.put("movs", 0);
   opcodes.put("movsx", 0);
   opcodes.put("movzx", 0);
   opcodes.put("mul", 0);
   opcodes.put("neg", 0);
   opcodes.put("nop", 0);
```

```
      opcodes.put("not", 0);
      opcodes.put("or", 0);
      opcodes.put("pop", 0);
      opcodes.put("push", 0);
      opcodes.put("ret", 0);
      opcodes.put("sub", 0);
      opcodes.put("test", 0);
      opcodes.put("xor", 0);
      opcodes.put("xchg", 0);
      //opcodes.put("dd", 0);
      opcodes.put("loc", 0);
      //opcodes.put("dll", 0);
      opcodes.put("jnb", 0);
      opcodes.put("jz", 0);
      opcodes.put("jnz", 0);
      opcodes.put("ja", 0);
      opcodes.put("jb", 0);
      opcodes.put("jl", 0);
      opcodes.put("js", 0);
      opcodes.put("shl", 0);
      opcodes.put("shr", 0);
   }

}
```

TestData.java

```
package opcode.test;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class TestData {

   private static HashMap<String, Integer> ngrams = new HashMap<String,
       Integer>();

   public static void main(String args[]) throws IOException{
      PrintWriter p;
      File f = new File("/home/vbox/test_asm.txt");
      BufferedReader br = new BufferedReader(new FileReader(f));
```

```
      p = new PrintWriter(new File("/u/tdshah/outputTestNgrams2.csv"));

      File finalOutput = new File("/u/tdshah/finaloutput2.txt");
      BufferedReader br2 = new BufferedReader(new FileReader(finalOutput));

      String line = "";
      while((line = br2.readLine())!= null){
         //System.out.println(line);
         int i = line.indexOf(":");
         line = line.substring(0, i).trim();
         System.out.println(line + "+");
         ngrams.put(line, 0);
      }

      String out = "Filename,";
      Iterator it = ngrams.entrySet().iterator();
      while (it.hasNext()) {
         Map.Entry pair = (Map.Entry) it.next();
         out += pair.getKey() + ",";
      }

      p.write(out +"\n");

      ExecutorService executor = Executors.newFixedThreadPool(500);
      int count = 0;

      line ="";
      while((line = br.readLine())!= null){
         int i = line.indexOf(".");
         line = line.substring(0, i);

         Runnable worker = new TestDataWorker(line, p, ngrams);
         executor.execute(worker);

         count++;

      }

      br.close();
      br2.close();
      executor.shutdown();
      while (!executor.isTerminated()) {
      }
      p.close();

   }

}
```

TestDataWorker.java

```
package opcode.test;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

public class TestDataWorker implements Runnable {

   private static int threadcount;
   private int threadName;
   private BufferedReader br;
   private HashMap<String, Integer> opcodes;
   private String filename;
   private PrintWriter p;
   private HashMap<String, Integer> ngrams;
   private File input;

   public TestDataWorker(String filename, PrintWriter p, HashMap<String,
       Integer> ngrams){

      threadcount++;
      threadName = threadcount;
      this.p = p;
      this.filename = filename;
      this.ngrams = new HashMap<String, Integer>(ngrams);
   }

   @Override
   public void run() {
      input = new File("/home/vbox/ngramFreq/" + filename + ".txt");
      System.out.println(threadName + " start");
      System.out.println(filename);
      try {
         br = new BufferedReader(new FileReader(input));
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      }

      String line = "";
      try {
         while((line = br.readLine())!= null){
```

```
            String s[] = line.split(":");
         // System.out.println(line);
         // System.out.println(s[0].trim());
            if(ngrams.containsKey(s[0].trim())){
            // System.out.println("Entered");
               ngrams.put(s[0].trim(), Integer.parseInt(s[1]));
            }
         }
      } catch (IOException e) {
         e.printStackTrace();
      }
      finally{
         try {
            br.close();
         } catch (IOException e) {
            e.printStackTrace();
         }

         String out = filename + ",";
         Iterator it = ngrams.entrySet().iterator();
         while (it.hasNext()) {
            Map.Entry pair = (Map.Entry) it.next();
            out += pair.getValue() + ",";
         }

         synchronized (p) {
            p.append(out + "\n");
         }
      }



      System.out.println(threadName + " end");
   }


}
```

MergeNgram.java

```
package opcode.ngram;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Iterator;
```

```
import java.util.Map;
import java.util.Scanner;

public class MergeNgram {

   private static HashMap<String, Integer> ngrams = new HashMap<String,
       Integer>();

   public static void main(String args[]) throws IOException {
      File f = new File("/u/tdshah/Downloads/trainLabels.txt");
      Scanner sc = new Scanner(f);
      File file = new File("/home/vbox/finaloutput2.txt");
      PrintWriter pr = new PrintWriter(file);
      int count = 0 ;
      /*while (sc.hasNext()) {

         String string = sc.nextLine();
         String x[] = string.split("\t");
         if (x[1].equals("9")) {

            File f2 = new File("/home/vbox/ngramFreq/" + x[0] + ".txt");
            BufferedReader br = new BufferedReader(new FileReader(f2));
            String line = "";

            while ((line = br.readLine()) != null) {
               String q[] = line.split(":");
               //System.out.println(q[0]);
               if (!ngrams.containsKey(q[0])) {
                  ngrams.put(q[0], 1);
               } else {
                  int p = ngrams.get(q[0]);
                  ngrams.put(q[0], ++p);
               }
            }
            count++;
         }

      }
*/
      for(int i=1; i<=9; i++){
         File f2 = new File("/home/vbox/RelevantNgrams2/outputclass" + i +
             ".txt");
         BufferedReader br = new BufferedReader(new FileReader(f2));
         String line = "";

         while ((line = br.readLine()) != null) {
            String q[] = line.split(":");
            System.out.println(q[0]);
            if (!ngrams.containsKey(q[0])) {
```

```
            ngrams.put(q[0], 1);
         } else {
            int p = ngrams.get(q[0]);
            ngrams.put(q[0], ++p);
         }
      }
   }

   System.out.println(count);
   String out = "";
   Iterator it = ngrams.entrySet().iterator();
   int qq = 0;
   int coun = 0;
   System.out.println(ngrams.size());
   while (it.hasNext()) {
      Map.Entry pair = (Map.Entry) it.next();
      System.out.println("check:"+coun);
      coun++;
      //if((Integer)pair.getValue() >= 450){
         qq++;
         out += pair.getKey() + ":" + pair.getValue() + "\n";
      //}
   }
   System.out.println(qq);
   pr.write(out);
   pr.close();
 }
}
```

B. RScript for feature selection

```
library(RMySQL)
install.packages(RMySQL)
library(corrplot)
#Connection to MySQL
con <- dbConnect(MySQL(), user="root", password="abcd", dbname="project", host="localhos
dbListTables(con, "project" )

#dbWriteTable(con,"tablename",datafram Name,overwrite=T)
rs = dbSendQuery(con, "select * from bytefrequency b, opcodefrequency o where b.Filename
d = fetch(rs, n=-1);

#final table for opcode and byte frequencies
#rs_freq = dbSendQuery(con, "select Filename,class,mov,extrn,cmp,jz,jnz,‘call‘,lea,jl,mo
#  inc,‘and‘,‘or‘,‘dec‘,‘leave‘,sub,test,shr,pop,
#  ‘xor‘,jnb,‘add‘,‘push‘,‘not‘,‘neg‘,jb,imul,mul,
#  ja,movsx,‘div‘,xchg,js,idiv,X133,X11,X7,X182,X15,X2,X19,X18,X6,X159,X131,X199,X125,X1
#  X155,X136,X151,X143,X157,X132,X139,X20,X195,X198,X153,X235,X128,X106,X147,X3,X94,X145
#  X17,X141,X8,X5,X169,X23,X24,X137,X150,X156,X89,X174,X1,X166,X154,X146,X233,X240,X30,X
#  X77,X135,X248,X86,X116,X85,X51,X175,X236,X134,X252,X26,X29,X32,X171,X162,X173,X104,X6
#  X87,X170,X59,X117,X70,X142,X72,X14,X22,X93,X152,X25,X254,X167,X183,X164,X31,X33,X163,
#  X202,X255,X46,X12,X57,X187,X80,X148,X242,X27,X123,X47,X191,X10,X205,X34,X165,X245,X4,
#  ,X213,X41,X185,X122,X42,X172,X68,X0,X119,X229,X168,X221,X223,X181,X35,X227,X13,X67,X2
#  from frequency;");


#rs_freq = dbSendQuery(con, "select class,mov,extrn,lea,jl,movzx,jmp,shl,inc,‘and‘,‘dec‘
rs_freq = dbSendQuery(con, "select * from frequency");
ds_train = fetch(rs_freq,n=-1);
dim(ds);
corrplot(cor(ds[,2:10]), method = 'number')

sub <- subset(ds, select = -c(cmp));

#finding correlation
ds_corr = ds[,2:186]
class(ds_corr)


corrplot(cor(ds_corr[,1:10]), method = 'number')

#randomforest Algoritdshm
install.packages("randomForest");
```

```
library(randomForest);
set.seed(1345);
data(ds);
ds.rf <- randomForest(as.factor(class)~.,data=ds,ntrees=170);
varImpPlot(ds.rf);



####
ds_train <- subset(ds_train, select=-c(row_names));

install.packages('rJava')
library(rJava)

install.packages("caret")
findCorrelation(corr, cutoff = .90, verbose = TRUE)

####################

############################
#f_ngram_train2[,2:2056] <- sapply(f_ngram_train2[,2:2056], as.numeric);
#corr = cor(f_ngram_train2[,2:2056])
#findCorrelation(corr, cutoff = .90, verbose = TRUE)
library(FSelector)
X = information.gain(class~., f_ngram_train2[,1:1000])
class = f_ngram_train[,2]
part = f_ngram_train2[,1001:2056]
combine <- cbind(class, part)
X2 = information.gain(class~., combine)
together <- rbind(X, X2)
sub <- cutoff.k(together, 1700)
Y = f_ngram_train[, c(sub)]
f_ngram_train3 = cbind(f_ngram_train[,1:2], Y)
```

C. Database Script

```
CREATE TABLE opcodefrequency ( filename VARCHAR(50), class VARCHAR(5),  ret VARCHAR(5),
‘neg‘ VARCHAR(5),
‘call‘ VARCHAR(5),
sub VARCHAR(5),
jcc VARCHAR(5),
bytes VARCHAR(5),
test VARCHAR(5),
js VARCHAR(5),
jnb VARCHAR(5),
‘div‘ VARCHAR(5),
ltr VARCHAR(5),
cmp VARCHAR(5),
pop VARCHAR(5),
jl VARCHAR(5),
‘add‘ VARCHAR(5),
push VARCHAR(5),
movs VARCHAR(5),
‘dec‘ VARCHAR(5),
inc VARCHAR(5),
‘leave‘ VARCHAR(5),
shr VARCHAR(5),
movzx VARCHAR(5),
jmp VARCHAR(5),
xchg VARCHAR(5),
shl VARCHAR(5),
jz VARCHAR(5),
jnz VARCHAR(5),
‘xor‘ VARCHAR(5),
iret VARCHAR(5),
‘or‘ VARCHAR(5),
idiv VARCHAR(5),
‘lock‘ VARCHAR(5),
loc VARCHAR(5),
extrn VARCHAR(5),
mul VARCHAR(5),
nop VARCHAR(5),
movsx VARCHAR(5),
‘and‘ VARCHAR(5),
‘not‘ VARCHAR(5),
mov VARCHAR(5),
jb VARCHAR(5),
imul VARCHAR(5),
```

```
lea VARCHAR(5),
ja VARCHAR(5),
bound VARCHAR(5));


select * from byteFrequency;


CREATE TABLE byteFrequency ( filename VARCHAR(50), class VARCHAR(5),  ‘0‘ VARCHAR(4),
‘1‘ VARCHAR(4),
‘2‘ VARCHAR(4),
‘3‘ VARCHAR(4),
‘4‘ VARCHAR(4),
‘5‘ VARCHAR(4),
‘6‘ VARCHAR(4),
‘7‘ VARCHAR(4),
‘8‘ VARCHAR(4),
‘9‘ VARCHAR(4),
‘10‘ VARCHAR(4),
‘11‘ VARCHAR(4),
‘12‘ VARCHAR(4),
‘13‘ VARCHAR(4),
‘14‘ VARCHAR(4),
‘15‘ VARCHAR(4),
‘16‘ VARCHAR(4),
‘17‘ VARCHAR(4),
‘18‘ VARCHAR(4),
‘19‘ VARCHAR(4),
‘20‘ VARCHAR(4),
‘21‘ VARCHAR(4),
‘22‘ VARCHAR(4),
‘23‘ VARCHAR(4),
‘24‘ VARCHAR(4),
‘25‘ VARCHAR(4),
‘26‘ VARCHAR(4),
‘27‘ VARCHAR(4),
‘28‘ VARCHAR(4),
‘29‘ VARCHAR(4),
‘30‘ VARCHAR(4),
‘31‘ VARCHAR(4),
‘32‘ VARCHAR(4),
‘33‘ VARCHAR(4),
‘34‘ VARCHAR(4),
‘35‘ VARCHAR(4),
```

```
`36` VARCHAR(4),
`37` VARCHAR(4),
`38` VARCHAR(4),
`39` VARCHAR(4),
`40` VARCHAR(4),
`41` VARCHAR(4),
`42` VARCHAR(4),
`43` VARCHAR(4),
`44` VARCHAR(4),
`45` VARCHAR(4),
`46` VARCHAR(4),
`47` VARCHAR(4),
`48` VARCHAR(4),
`49` VARCHAR(4),
`50` VARCHAR(4),
`51` VARCHAR(4),
`52` VARCHAR(4),
`53` VARCHAR(4),
`54` VARCHAR(4),
`55` VARCHAR(4),
`56` VARCHAR(4),
`57` VARCHAR(4),
`58` VARCHAR(4),
`59` VARCHAR(4),
`60` VARCHAR(4),
`61` VARCHAR(4),
`62` VARCHAR(4),
`63` VARCHAR(4),
`64` VARCHAR(4),
`65` VARCHAR(4),
`66` VARCHAR(4),
`67` VARCHAR(4),
`68` VARCHAR(4),
`69` VARCHAR(4),
`70` VARCHAR(4),
`71` VARCHAR(4),
`72` VARCHAR(4),
`73` VARCHAR(4),
`74` VARCHAR(4),
`75` VARCHAR(4),
`76` VARCHAR(4),
`77` VARCHAR(4),
`78` VARCHAR(4),
`79` VARCHAR(4),
```

```
‘80‘ VARCHAR(4),
‘81‘ VARCHAR(4),
‘82‘ VARCHAR(4),
‘83‘ VARCHAR(4),
‘84‘ VARCHAR(4),
‘85‘ VARCHAR(4),
‘86‘ VARCHAR(4),
‘87‘ VARCHAR(4),
‘88‘ VARCHAR(4),
‘89‘ VARCHAR(4),
‘90‘ VARCHAR(4),
‘91‘ VARCHAR(4),
‘92‘ VARCHAR(4),
‘93‘ VARCHAR(4),
‘94‘ VARCHAR(4),
‘95‘ VARCHAR(4),
‘96‘ VARCHAR(4),
‘97‘ VARCHAR(4),
‘98‘ VARCHAR(4),
‘99‘ VARCHAR(4),
‘100‘ VARCHAR(4),
‘101‘ VARCHAR(4),
‘102‘ VARCHAR(4),
‘103‘ VARCHAR(4),
‘104‘ VARCHAR(4),
‘105‘ VARCHAR(4),
‘106‘ VARCHAR(4),
‘107‘ VARCHAR(4),
‘108‘ VARCHAR(4),
‘109‘ VARCHAR(4),
‘110‘ VARCHAR(4),
‘111‘ VARCHAR(4),
‘112‘ VARCHAR(4),
‘113‘ VARCHAR(4),
‘114‘ VARCHAR(4),
‘115‘ VARCHAR(4),
‘116‘ VARCHAR(4),
‘117‘ VARCHAR(4),
‘118‘ VARCHAR(4),
‘119‘ VARCHAR(4),
‘120‘ VARCHAR(4),
‘121‘ VARCHAR(4),
‘122‘ VARCHAR(4),
‘123‘ VARCHAR(4),
```

```
`124` VARCHAR(4),
`125` VARCHAR(4),
`126` VARCHAR(4),
`127` VARCHAR(4),
`128` VARCHAR(4),
`129` VARCHAR(4),
`130` VARCHAR(4),
`131` VARCHAR(4),
`132` VARCHAR(4),
`133` VARCHAR(4),
`134` VARCHAR(4),
`135` VARCHAR(4),
`136` VARCHAR(4),
`137` VARCHAR(4),
`138` VARCHAR(4),
`139` VARCHAR(4),
`140` VARCHAR(4),
`141` VARCHAR(4),
`142` VARCHAR(4),
`143` VARCHAR(4),
`144` VARCHAR(4),
`145` VARCHAR(4),
`146` VARCHAR(4),
`147` VARCHAR(4),
`148` VARCHAR(4),
`149` VARCHAR(4),
`150` VARCHAR(4),
`151` VARCHAR(4),
`152` VARCHAR(4),
`153` VARCHAR(4),
`154` VARCHAR(4),
`155` VARCHAR(4),
`156` VARCHAR(4),
`157` VARCHAR(4),
`158` VARCHAR(4),
`159` VARCHAR(4),
`160` VARCHAR(4),
`161` VARCHAR(4),
`162` VARCHAR(4),
`163` VARCHAR(4),
`164` VARCHAR(4),
`165` VARCHAR(4),
`166` VARCHAR(4),
`167` VARCHAR(4),
```

```
`168` VARCHAR(4),
`169` VARCHAR(4),
`170` VARCHAR(4),
`171` VARCHAR(4),
`172` VARCHAR(4),
`173` VARCHAR(4),
`174` VARCHAR(4),
`175` VARCHAR(4),
`176` VARCHAR(4),
`177` VARCHAR(4),
`178` VARCHAR(4),
`179` VARCHAR(4),
`180` VARCHAR(4),
`181` VARCHAR(4),
`182` VARCHAR(4),
`183` VARCHAR(4),
`184` VARCHAR(4),
`185` VARCHAR(4),
`186` VARCHAR(4),
`187` VARCHAR(4),
`188` VARCHAR(4),
`189` VARCHAR(4),
`190` VARCHAR(4),
`191` VARCHAR(4),
`192` VARCHAR(4),
`193` VARCHAR(4),
`194` VARCHAR(4),
`195` VARCHAR(4),
`196` VARCHAR(4),
`197` VARCHAR(4),
`198` VARCHAR(4),
`199` VARCHAR(4),
`200` VARCHAR(4),
`201` VARCHAR(4),
`202` VARCHAR(4),
`203` VARCHAR(4),
`204` VARCHAR(4),
`205` VARCHAR(4),
`206` VARCHAR(4),
`207` VARCHAR(4),
`208` VARCHAR(4),
`209` VARCHAR(4),
`210` VARCHAR(4),
`211` VARCHAR(4),
```

```
`212` VARCHAR(4),
`213` VARCHAR(4),
`214` VARCHAR(4),
`215` VARCHAR(4),
`216` VARCHAR(4),
`217` VARCHAR(4),
`218` VARCHAR(4),
`219` VARCHAR(4),
`220` VARCHAR(4),
`221` VARCHAR(4),
`222` VARCHAR(4),
`223` VARCHAR(4),
`224` VARCHAR(4),
`225` VARCHAR(4),
`226` VARCHAR(4),
`227` VARCHAR(4),
`228` VARCHAR(4),
`229` VARCHAR(4),
`230` VARCHAR(4),
`231` VARCHAR(4),
`232` VARCHAR(4),
`233` VARCHAR(4),
`234` VARCHAR(4),
`235` VARCHAR(4),
`236` VARCHAR(4),
`237` VARCHAR(4),
`238` VARCHAR(4),
`239` VARCHAR(4),
`240` VARCHAR(4),
`241` VARCHAR(4),
`242` VARCHAR(4),
`243` VARCHAR(4),
`244` VARCHAR(4),
`245` VARCHAR(4),
`246` VARCHAR(4),
`247` VARCHAR(4),
`248` VARCHAR(4),
`249` VARCHAR(4),
`250` VARCHAR(4),
`251` VARCHAR(4),
`252` VARCHAR(4),
`253` VARCHAR(4),
`254` VARCHAR(4),
`255` VARCHAR(4),
```

```
‘256‘ VARCHAR(4));


LOAD DATA LOCAL INFILE ’C:\\Users\\Teja\\Masters\\DataMining\\Data\\bytefreq.csv’
INTO TABLE bytefrequency
FIELDS TERMINATED BY ’,’
ENCLOSED BY ’"’
LINES TERMINATED BY ’\n’
IGNORE 1 ROWS;

select count(*) from bytefrequency;
truncate table bytefrequency;

select Filename from bytefrequency;

select * from opcodefrequency;

create index idx_filename on opcodefrequency (FileName);

desc bytefrequency;


ALTER TABLE opcodefrequency MODIFY FileName VARCHAR(30);

ALTER TABLE bytefrequency ADD PRIMARY KEY(filname);

ALTER TABLE opcodefrequency ADD PRIMARY KEY(FileName);

select * from frequency;



select class,mov,extrn,cmp,jz,jnz,‘call‘,lea,jl,movzx,jmp,shl,inc,‘and‘,‘or‘,‘dec‘,‘leav
,‘xor‘,jnb,‘add‘,‘push‘,‘not‘,‘neg‘,jb,imul,mul,ja,movsx,‘div‘,xchg,js,idiv,X133,X11,X7,
X19,X18,X6,X159,X131,X199,X125,X192,X155,X136,X151,X143,X157,X132,X139,X20,X195,X198,X15
X106,X147,X3,X94,X145,X196,X179,X16,X17,X141,X8,X5,X169,X23,X24,X137,X150,X156,X89,X174,
,X146,X233,X240,X30,X158,X232,X130,X38,X77,X135,X248,X86,X116,X85,X51,X175,X236,X134,X25
2,X171,X162,X173,X104,X69,X177,X9,X64,X83,X87,X170,X59,X117,X70,X142,X72,X14,X22,X93,X15
167,X183,X164,X31,X33,X163,X149,X127,X237,X178,X202,X255,X46,X12,X57,X187,X80,X148,X242,
,X191,X10,X205,X34,X165,X245,X4,X193,X186,X215,X197,X213,X41,X185,X122,X42,X172,X68,X0,X
8,X221,X223,X181,X35,X227,X13,X67,X21,X238,X160,X250,X39 from frequecy;
```

```
select * from bytefrequency b, opcodefrequency o, final_features_train_ngram n
where b.Filename = o.filename and b.FileName=n.FileName;

select * from bytefrequency b, opcodefrequency o where b.Filename = o.filename;
```

D. Python scripts

nGram.py

```
import sklearn
import os
from sklearn.ensemble import RandomForestClassifier
import csv_io
import scipy

def main():
    f = open("random_forest_solution","w")
    #read in the training file
    train = csv_io.read_data("freq_train_29Apr -py.csv")
    #set the training responses
    target = [x[0] for x in train]
    #set the training features
    train = [x[1:] for x in train]
    #read in the test file
    realtest = csv_io.read_data("freq_test_29Apr -py.csv")

    # random forest code
    rf = RandomForestClassifier(n_estimators=150, min_samples_split=2,
        n_jobs=-1)
    # fit the training data
    print('fitting the model')
    rf.fit(train, target)
    # run model against test data
    predicted_probs = rf.predict_proba(realtest)
    #print(type(predicted_probs))
    #print(predicted_probs)
    count=0
    for arr in predicted_probs:
        count+=1
        row=[1.0]
        row.clear()
        for val in arr:
            row.append(val)
        #print(row)
        for i in range(0,len(row)):
            f.write(str(row[i]))
            if(i!=len(row)-1):
                f.write(",")
        f.write("\n")
    f.close()
    print(count)
```

```
    #predicted_probs = ["%f" % x[1] for x in predicted_probs]
    #csv_io.write_delimited_file("random_forest_solution.csv", predicted_probs)

    print ('Solution generated')

os.umask(0000)
os.chdir('/home/manu/kaggle')
main()
```

RandomForest.py

```
import sklearn
import os
from sklearn.ensemble import RandomForestClassifier
import csv_io
import scipy

def main():
    f = open("random_forest_solution","w")
    #read in the training file
    train = csv_io.read_data("freq_train_29Apr -py.csv")
    #set the training responses
    target = [x[0] for x in train]
    #set the training features
    train = [x[1:] for x in train]
    #read in the test file
    realtest = csv_io.read_data("freq_test_29Apr -py.csv")

    # random forest code
    rf = RandomForestClassifier(n_estimators=150, min_samples_split=2,
        n_jobs=-1)
    # fit the training data
    print('fitting the model')
    rf.fit(train, target)
    # run model against test data
    predicted_probs = rf.predict_proba(realtest)
    #print(type(predicted_probs))
    #print(predicted_probs)
    count=0
    for arr in predicted_probs:
        count+=1
        row=[1.0]
        row.clear()
        for val in arr:
            row.append(val)
```

```
        #print(row)
        for i in range(0,len(row)):
            f.write(str(row[i]))
            if(i!=len(row)-1):
                f.write(",")
        f.write("\n")
    f.close()
    print(count)









    #predicted_probs = ["%f" % x[1] for x in predicted_probs]
    #csv_io.write_delimited_file("random_forest_solution.csv", predicted_probs)

    print ('Solution generated')

os.umask(0000)
os.chdir('/home/manu/kaggle')
main()
```