Spring 2021

Hidden Markov Model-based Clustering for Malware Classification

Shamli Singh
San Jose State University

Hidden Markov Model-based Clustering for Malware Classification

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Shamli Singh

May 2021

The Designated Project Committee Approves the Project Titled

Hidden Markov Model-based Clustering for Malware Classification

by

Shamli Singh

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2021

Dr. Mark Stamp	Department of Computer Science
Dr. William Andreopoulos	Department of Computer Science
Dr. Fabio Di Troia	Department of Computer Science

ABSTRACT

Hidden Markov Model-based Clustering for Malware Classification

by Shamli Singh

Automated techniques to classify malware samples into their respective families are critical in cybersecurity. Previously research applied k-means clustering to scores generated by hidden Markov models (HMM) as a means of dealing with the malware classification problem. In this research, we follow a somewhat similar approach, but instead of using HMMs to generate scores, we directly cluster the HMMs themselves. We obtain good results on a challenging malware dataset.

ACKNOWLEDGMENTS

I want to express my gratitude to my project advisor, Dr. Mark Stamp, for his guidance, support, and encouragement throughout my graduate studies. He has always been patient in listening to the tiniest issues and roadblocks that came up while working on the project.

I would also like to thank my committee member Dr. William Andreopoulos for his time and guidance.

I am further thankful to my committee member, Prof. Fabio Di Troia, for his valuable inputs and my senior, Sunhera Paul, to help with the dataset.

Finally, I would like to thank my parents and friends for their unwavering support and guidance throughout my Master's degree program.

TABLE OF CONTENTS

CHAPTER

APPENDIX

CHAPTER 1

Introduction

Malware, or malicious software, is a prominent antagonist in the digital era. We rely heavily on software for various aspects of civilization, from national security to trading, science research, or just entertainment. Analogously, malware plays a leading role in computer crime and information warfare, including malicious access of resources, stealing sensitive data, or simply spreading corrupt files through various computers. In recent years, smartphones are taking over desktop and laptop computers as the preferred device for consumers. Their popularity and constant internet connectivity mean an increase in the number of devices susceptible to malware attacks. According to [1], malware is spread rapidly, and it continuously evolves to escape malware detection tools. Usually, malware detection tools depend on malware signatures. However, this can be slow, and new malware variants cannot be detected.

Malware can appear in various forms, including viruses, spyware, and worms [2, 3]. Malware is a critically important research topic in cybersecurity. Many malware detection techniques are available, and we elaborate on some of these in Chapter 2.

Malware attacks come with a lot of costs, both visible and hidden. The visible costs include the time and money required to deal with malware detection and clean up infected systems. It also includes the loss of productivity in the affected organization. The damage to the organization's reputation represents the hidden costs, as observed in [2].

There is still a lack of robust strategies to detect or classify malware. Malware classification is an essential aspect of protecting computer systems. If we can determine the family that a new malware belongs to, we can save time finding the correct strategy to nullify it by using known techniques. On the other hand, if detected malware does not fit the existing types, then we need to find new approaches.

In this paper, we focus on automated malware classification using hidden Markov models (HMMs) and clustering techniques. Specifically, we train HMMs on malware opcodes representing various families and then perform clustering on these models to classify them into their respective families. This work can be viewed as an extension of the previous research in [4].

The remainder of this paper is organized as follows: Chapter 2 includes relevant background information on malware detection strategies, and also discusses HMM and clustering in detail. Chapter 3 gives an overview of related previous work on malware classification. Chapter 4 covers the implementation of clustering algorithms used for malware classification in this research, as well as our experiments and results. Finally, Chapter 5 concludes with suggested future work.

CHAPTER 2

Background

2.1 Malware

Our computer infrastructure is at constant war with various forms of threats like bugs, spam, denial of service (DoS) attacks, and malware [2]. As discussed in Chapter 1, malware dominates all security threats. The purpose of writing malware can be simply a prank or as threatening as organized crimes or warfare or espionage. Figure 1 shows the rapid increase in the volume of malware attacks for the past decade.

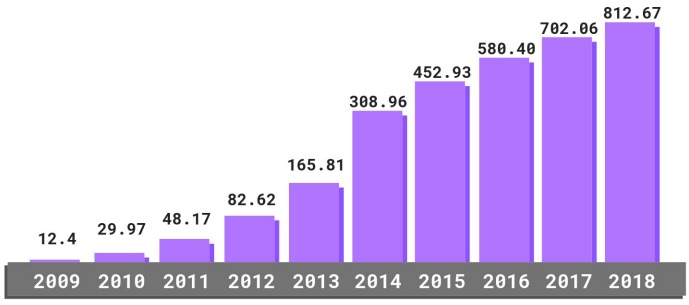

Figure 1: Total Malware Infection Growth Rate [5]

Malware covers a broad spectrum of threats like viruses, worms, and spyware. We give a brief overview of some of them before moving to malware detection techniques.

2.1.1 Virus

A virus is the most commonly heard type of malware. True to its name, this malware replicates itself by infecting executable programs. The infected programs further propagate the virus during their execution. It can also live in the computer memory and propagates through external devices or software, or emails. Like a biological virus, it may also exhibit metamorphism – it can change its form while infecting more programs [3]. Malware writers explicitly use this feature to evade malware detection tools.

2.1.2 Trojan Horse

Like the historic plot by Greek invaders to capture Troy, a Trojan Horse is a program devised to look harmless by performing a benign task and secretly perform a malicious task. Password and form grabbers fall under this category, as described by [2]. According to [6], they can also control a user's system or disable the firewall. They are also capable of downloading other malware to the system. Zeus or Zbot is a famous trojan family, and it was widely used for crimes like bank frauds and money laundering in the last two decades [7].

2.1.3 Backdoor

A backdoor, also known as a trapdoor, is built to circumvent a standard security check [2]. Programmers may create backdoors for legitimate reasons when their code is in the development phase. Still, cybercriminals exploit this vulnerability for tasks like deleting files, accessing sensitive data, installing additional malware, and opening communication ports for remote access of the system [8].

2.1.4 Spyware and Adware

Spyware is a program that installs itself on a computer system along with some benign software and spy on user activities, including internet usage history, keystrokes, forms, and also read files covertly and install additional malware [9]. Adware is slightly different from spyware. Adware looks like a regular program but instead pops up errors or updates and then asks the user to pay money to continue working. Winwebsec is a famous family of adware [10].

2.2 Malware Detection Techniques

Malware writers constantly improve their malware by making it undetectable for signature-based antivirus tools. Similarly, security software companies continue their research on ways to improve their malware detection techniques. The following

sections describe the strategies used in building these techniques.

2.2.1 Signature Based Detection

Signature-based detection is the prevalent technique deployed by many antivirus software due to its accuracy, speed, and simplicity [3]. The software scans each executable for malware signature or pattern, which is generally a string of bits and kept in the antivirus database. Despite being popular, the downside to this approach is that antivirus software can detect only known malware based on its updated database. Malicious users generally evade these tools by obfuscating the malware code.

2.2.2 Anomaly Based Detection

Anomaly detection is designed to find unusual and potentially malicious activity. It is a heuristic approach to training the system with normal behavior [11]. The system can then flag anything outside the expected behavior. While this technique can help detect previously unknown malware, it is not yet proven robust enough as a standalone technique and involves false positives [3].

2.2.3 Machine Learning Based Detection

Anomaly-based detection above can be categorized as a machine learning technique and a behavioral technique. But pure machine learning techniques are also adopted for malware detection and classification. Researchers in [12] tested this approach by computing behavior scores in a sandbox and using them to generate sparse vector models. They used these models with various machine learning techniques like Support Vector Machine (SVM), naïve Bayes, decision trees, random forest, and multilayer perceptron (MLP). Another research suggested using AutoEncoder, a deep learning technique for malware detection [13].

2.2.4 Hidden Markov Model Based Detection

Hidden Markov models (HMMs) are used for statistical pattern analysis, broadly in speech recognition, biological sequence analysis, and malware detection [14]. The general approach tests the malware dataset against trained HMMs and computes scores for each file in the dataset. The malware and benign scores will not overlap for a range of values known as a threshold [15]. This threshold is then used to distinguish benign files from malignant files.

2.3 Hidden Markov Models

A statistical model with states and known probabilities of the state transitions is called a Markov model [15]. The states are visible to an observer in such a model. On the contrary, a hidden Markov model has states that are not directly observable [14]. HMM acts as a state machine, where each state is associated with a probability distribution for observing a state of observation symbols. We can train an HMM using the observation sequences that represent some data.

2.3.1 Notation

As given in [15], we use the notation given in Table 1 to define an HMM.

Table 1: HMM Notation

T = length of observation sequence
N = number of states in the model
M = number of distinct observation symbols
Q = distinct states in the model
V = set of possible observations
A = state transition probability matrix
B = observation probability matrix
π = initial state distribution
$\mathcal{O}$ = observation sequence

An HMM is defined by the A, B and π matrices, and is denoted as $\lambda = (A, B, \pi)$. Figure 2 shows a generic HMM representation.

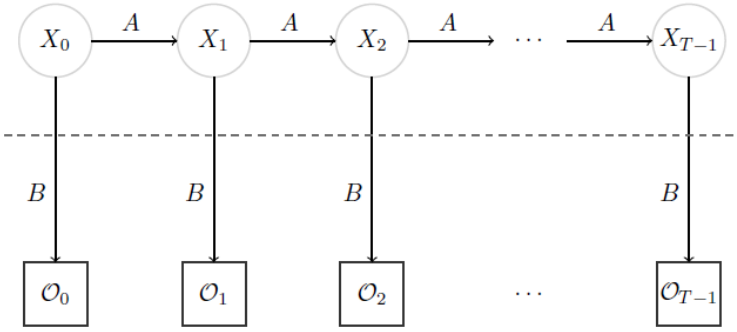

Figure 2: A Hidden Markov Model [16]

2.3.2 The Three Problems

Three fundamental problems can be solved using an HMM. They are described below.

2.3.2.1 Problem 1

Given a model $\lambda = (A, B, \pi)$ and an observation sequence $\mathcal{O}$, we need to find $P(\mathcal{O} \mid \lambda)$. This means we want to compute a score for an observed sequence $\mathcal{O}$ with respect to a model λ [15].

2.3.2.2 Problem 2

Given a model $\lambda = (A, B, \pi)$ and an observation sequence $\mathcal{O}$, we want to reveal the hidden part of the model [15].

2.3.2.3 Problem 3

Given an observation sequence $\mathcal{O}$ and the number of states N, we need to determine the model $\lambda = (A, B, \pi)$ that maximizes the probability of $\mathcal{O}$. This also implies we train a model to fit the observed data, using a hill-climb approach [15].

In this project, we are using the algorithm for Problem 3. We are using an observation sequence $\mathcal{O}$ with some number of states N and generating the models $\lambda = (A, B, \pi)$ for each malware opcode sequence file.

2.3.3 The Solutions

The problems are solved using the following algorithms:

2.3.3.1 The Forward Algorithm

Also known as the α pass, the forward algorithm is used to determine $P(\mathcal{O} \mid \lambda)$, as in problem 1. It is given as follows. For $t = 0, 1, ..., T-1$ and $i = 0, 1, \ldots, N-1$, define

$$\alpha_t(i) = P(\mathcal{O}_0, \mathcal{O}_1, \ldots, \mathcal{O}_t, X_t = q_t|\lambda)$$

where $\alpha_t(i)$ is the probability of partial observation sequence up to time t.

Thus, $P(\mathcal{O} \mid \lambda)$ can be computed as follows.

1. Let $\alpha_t(i) = \pi_t b_i(\mathcal{O}_0)$, for $i = 0, 1, \ldots, N-1$

2. For $t = 0, 1, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, compute

$$\alpha_t(i) = \left(\sum_{j=1}^{N-1} \alpha_{t-1}(j) a_{ij} \right) b_i(\mathcal{O}_t)$$

2.3.3.2 The Backward Algorithm

Also known as the β pass, the backward algorithm is used to find a most likely optimal state sequence, like in problem 2. It is given as follows: For $t = 0, 1, \ldots, T-1$ and $i = 0, 1, \ldots, N-1$, define

$$\beta_t(i) = P(\mathcal{O}_{t+1}, \mathcal{O}_{t+2}, \ldots, \mathcal{O}_{T-1}, X_t = q_t|\lambda)$$

Thus, $\beta_t(i)$ can be computed as follows.

1. Let $\beta_t(i) = 1$, for $i = 0, 1, \ldots, N-1$

2. For $t = T-2, T-3, \ldots, 0$ and $i = 0, 1, \ldots, N-1$, compute

$$\beta_t(i) = \sum_{j=0}^{N-1} a_{ij} b_j(\mathcal{O}_{t+1}) \beta_{t+1}(j)$$

For $t = 0, 1, \ldots, T-2$ and $i = 0, 1, \ldots, N-1$, define

$$\gamma_t(i) = P(x_t = q_i | \mathcal{O}, \lambda)$$

The relevant probability up to time t is given by

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(\mathcal{O} \mid \lambda)}$$

2.3.3.3 The Baum-Welch Re-estimation Algorithm

The Baum-Welch algorithm helps to re-estimate the A, B and π matrices iteratively [15]. This helps efficiently fit a model to observations, according to problem 3. The parameters N and M remain constant throughout the process, whereas the A, B and π matrices can change with the row-stochastic condition in place. The process is as follows.

1. Initialize $\lambda = (A, B, \pi)$ with random values. Generally these are set as $\pi = 1/N$, $A_{ij} = 1/N$ and $B_{ij} = 1/M$.

2. Compute $\alpha_t(i)$, $\beta_t(i)$, $\gamma_t(i)$ and $\gamma_t(i, j)$ where $\gamma_t(i, j)$ is a di-gamma. It is defined as

$$\gamma_t(i) = \frac{\alpha_t(i)a_{ij}b_j(\mathcal{O}_{t+1})\beta_{t+1}(j)}{P(\mathcal{O} \mid \lambda)}$$

   where

$$\gamma_t(i) = \sum_{j=0}^{N-1} \gamma_t(i, j)$$

3. Re-estimate model parameters For $i = 0, 1, \ldots, N-1$, let

$$\pi_i = \gamma_0(i)$$

   For $i = 0, 1, \ldots, N-1$ and $j = 0, 1, \ldots, N-1$, compute

$$a_{ij} = \frac{\sum_{t=0}^{T-2} \gamma_t(i, j)}{\sum_{t=0}^{T-2} \gamma_t(i)}$$

For $j = 0, 1, \ldots, N-1$ and $k = 0, 1, \ldots, M-1$, compute

$$b_{ij} = \frac{\sum\limits_{\substack{t \in \{0,1\ldots,T-1\} \\ \mathcal{O}_t = k}} \gamma_t(j)}{\sum\limits_{t=0}^{T-2} \gamma_t(j)}$$

4. If $P(\mathcal{O} \mid \lambda)$ increases, go back to step 3.

2.4 Clustering

Clustering is the process of categorizing objects into subsets based on some form of similarity index [17]. It is an unsupervised machine learning approach. Following are the most popular approaches to clustering objects.

- **Hierarchical Clustering:** The data is broken into a hierarchy of clusters. Each object is a separate cluster at the start. Then the algorithm computes a dendrogram by merging smaller clusters into larger ones (agglomerative approach) or dividing larger clusters into smaller ones (divisive approach) [18, 19]. The number of clusters is not predetermined, and the process only requires the similarity measure as an input.
- **Partitional Clustering:** Partitional clustering starts by first generating partitions using randomness or some policy and then evaluate them based on a criterion [17]. Each object is placed in one of k mutually exclusive divisions. Then the partition boundaries and centroids of the partitions change according to some algorithm [18]. K-means is one of the most popular partitional clustering algorithms, where the number of partitions k is predetermined. While this method is faster than hierarchical clustering, the output depends heavily on the choice of the number k.

2.4.1 K-means Clustering Algorithm

K-means is the most straightforward unsupervised learning algorithm used to solve classification problems using clustering [16]. It classifies data into a predetermined

number of clusters, called k. Cluster centroids are defined for each cluster, and each data point (or object) is associated with the centroid nearest to it [17]. The centroids may or may not be from within the dataset. Euclidean distance method is used. In the next step, the centroids are calculated again to account for the data points assigned to them. These steps keep repeating until there is no change to the centroid [17].

The Euclidean distance between two points $x = (x_1, x_2, x_3, \ldots, x_n)$ and $y = (y_1, y_2, y_3, \ldots, y_n)$ is given by

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2 + \cdots + (x_n - y_n)^2}$$

2.4.2 K-medoids Clustering Algorithm

$K-$medoids is a relatively popular method of clustering that searches for k objects called medoids. The medoids are considered representatives of the dataset as they are chosen from within the dataset. Like k-means, this algorithm minimizes the average dissimilarity of all data objects with the centroids, also known as medoids [20]. We can tune the algorithm to work with different types of similarity metrics like Euclidean distance, Minkowski distance, or cosine similarity, making it robust to noise and outliers. We can also feed just the set of dissimilarities instead of actual data objects to this algorithm. One way to do this is a distance correlation matrix, which is discussed in detail in 4. The algorithm itself is based on the Partitioning Around Medoids (PAM) program, as developed by the authors in [20].

2.4.3 Cluster Quality

An ideal cluster has minimal intra-cluster distance and maximum inter-cluster distance [16]. Following two methods are the standards used for internal and external validation of any clustering algorithm.

2.4.3.1 Elbow Method

The elbow method is a simple but not robust method of estimating the optimal number of k. The technique involves running the algorithm multiple times with an increasing number of clusters and plotting the clustering score (also known as distortion) against the number of clusters. The distortion is typically is the error sum of squares (SSE) [21]. As the number of clusters k increases, the SSE decreases. Sometimes we may not have clearly clustered data, so the elbow may not be sharp and clear.

2.4.3.2 Silhouette Analysis

The silhouette value is a measure of similarity of a data point to its own cluster compared to other clusters [21]. It ranges from -1 to 1, and a higher value indicates ideal clustering.

Silhouette value $s(i)$ for each point i is given as:

$$s(i) = \frac{b(i) - a(i)}{\max\Big(a(i), b(i)\Big)}$$

Here $a(i)$ is the measure of similarity of point i to its own cluster, also called cohesion. Similarly, $b(i)$ is the dissimilarity of point i from points in other clusters, also called as separation.

For each point i in the cluster C_i,

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \epsilon C_i, i \neq j} d(i, j)$$

and

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \epsilon C_k} d(i, j)$$

where $d(i, j)$ is the Euclidean distance between points i and j.

CHAPTER 3

Related Work

There has been much research in the domain of malware detection and classification already. This chapter will discuss some of the previous attempts at malware classification, and we can broadly categorize them as machine learning-based and characteristics-based.

3.1 Classification using Structure and Behavior of Malware

The earliest malware detection strategies have relied on the visible characteristics of malware, like their structure or behavior. These characteristics are usually dependent on how well the malware code is written and whether code obfuscation has been used. The strategies are further discussed in detail below.

3.1.1 Structure Control Flow

A control flow graph is the representation of the execution path a program might take. The research demonstrated in [22] successfully classified malware by using approximate matching of control flow graphs. They first analyze the entropy of malware binary to check if it has undergone code packing transformation. If packed, application-level emulation reveals hidden code on unpacking. String analysis then builds signatures for the control flow graphs. These signatures are compared with the existing malware database using string edit distances. A similarity between flow graphs is obtained and categorized accordingly based on a threshold value. The research has also worked successfully on variants of known malware.

3.1.2 Behavioral Malware Classification

To build more potent malware, malicious users sometimes make variants of existing malware using code obfuscation, thus confusing antivirus tools that depend on malware signatures. Even though malware variants of the same family have different signatures, they have common characteristic behavioral patterns due to their shared

heritage and function, as suspected by the authors of [23]. They built an automatic classification system trained on observed similarities in behavioral features extracted from malware variants in a family and then used decision trees to classify the malware.

3.2 Malware Classification using Machine Learning techniques

There have been various advances in the machine learning field, and malware detection and classification have not been held back in these advances. In [16], the author has demonstrated several applications of different machine learning techniques to real-world problems like malware detection and classification.

3.2.1 Support Vector Machines (SVM)

Support Vector Machines [16] are supervised machine learning models used for classification that can analyze and recognize patterns. SVMs use kernel function to map training data into a high-dimensional space and makes the problem linearly separable. After training, the model can classify new data into one of the categories. The research combines each malware scoring technique with SVM: HMMs, simple substitution distance (SSD), and opcode graph similarity (OGS) [24]. SVM was applied on scores obtained from each of these methods. They found the results to be more robust than using the individual techniques.

3.2.2 Naïve Bayes

Naïve Bayes classifier is a probabilistic method that has been used for a long time in various information retrieval and text classification systems [25]. The classifier stores the prior probability of each class of malware, $P(C_i)$, and the conditional probability of each attribute value given a class, $P(v_i|C_i)$ [16]. These probabilities are determined by training the classifier on the dataset. Then Bayes rule is used to compute the posterior probability of each class given an unknown instance and return its predicted class of malware.

3.2.3 Random Forest

A random forest is a machine learning method used for classification and regression. It is an ensemble of decision trees, where each tree determines the class label of an unlabeled instance [11]. Each tree is divided at each node, taking into account random features. In the end, the model selects the most chosen class amongst all trees. More number of trees can mean better accuracy, but random forest also tends to overfit on data [16].

3.2.4 Nearest Neighbor using VILO

In [26], researchers talk about VILO, a rapid-learning nearest neighbor algorithm for familial malware classification problem. VILO is a three-fold technique. First, N-perm is a robust version of n-gram of size n over malware opcodes. TF-IDF or term frequency and inverse document frequency is the vectorization method used to weigh malware features. TF-IDF makes the technique efficient for variants of known malware. Finally, the nearest neighbor algorithm is used on these feature vectors and feature weights to classify malware according to their families.

3.2.5 K-means Clustering with HMM

One proposed approach of malware classification is using clustering algorithms, which is the basis of research in [4]. The authors have coupled k-means clustering with HMM. First, the malware dataset was divided into 5 parts, out of which 4 parts were used to train HMMs, and then the fifth part scored against these models. The score is calculated as the log likelihood per opcode (LLPO) in a malware. They repeated the entire process for 7 different malware compilers so that each malware sample had seven scores, forming a tuple. These tuples were then used as the dataset for k-means clustering with experiments ranging from $k = 2$ to $k = 15$.

CHAPTER 4

Experiments and Results

4.1 Dataset

The list of malware families used in the experiments can be found in Table 2. Opcodes were previously extracted from the malware families and used as input for the experiments in this project. There are total 5,997 such files which are unequally distributed amongst the malware families.

Table 2: Malware Distribution in the dataset

Family	Samples
adload	56
bho	135
bifrose	305
buzus	6
ceeinject	665
delfinject	278
dorkbot	145
hotbar	8
hupigon	169
ircbot	34
obfuscator	272
rbot	34
vbinject	1342
vobfus	655
winwebsec	1160
zbot	322
zegost	411
Total	5997

4.2 Setup

In this project, we had opcodes data from previously disassembled malware files. All other experiments are performed on a single host machine. Following are the specifications of the host machine:

- Model: Lenovo Yoga 730

- Processor: Intel Core i7-8550U CPU @ 1.8 Ghz, 2.0 Ghz
- RAM: 16.0 GB
- Operating System: Windows 10 Home 64-bit

4.3 Training HMMs

As the whole project is based on finding distance between the HMMs, the training phase involves building HMMs on separate files in the dataset. For the experiments, we used all of the files mentioned above for training HMMs. The HMMs were trained on the opcode sequences in each of these files. The opcodes are treated as observation symbols for the HMM. We set the length of the observation sequence $T = 10,000$ throughout the experiments. Each HMM is trained for up to 50 iterations, and the model is generated. We repeat this process for a varying number of states N, from 2 to 4. All the models, denoted as $\lambda = (A, B, \pi)$ [15], are stored as individual comma-separated values (CSV) files for future use.

4.4 Clustering Algorithms

Instead of training HMMs on some of the dataset and scoring test data on these HMMs, we directly compare the HMMs and use the comparison for clustering them into respective families.

4.4.1 The Naïve K-means Method

We create a single CSV file with each malware opcode file denoted as a row, followed by the A and B matrices. To achieve this, we convert the $n \times n$ A matrix to a $1 \times n^2$ vector and $n \times m$ B matrix to a $1 \times nm$ vector and then store them to the CSV. Then we use these $n^2 + nm$ columns as the features in the k-means clustering algorithm. Since k-means depends on Euclidean distances between data points, we intrinsically compute distances between the models. Here we do not account for when the states would have flipped in the HMMs.

The following example can demonstrate this. Consider the HMMs for two malware files given by $\lambda_1 = (A_1, B_1, \pi_1)$ and $\lambda_2 = (A_2, B_2, \pi_2)$. Let $N = 2$ and $M = 30$. Thus we can give the matrices as follows:

$$A_1 = \begin{bmatrix} a1_{11} & a1_{12} \\ a1_{21} & a1_{22} \end{bmatrix} \tag{1}$$

and

$$A_2 = \begin{bmatrix} a2_{11} & a2_{12} \\ a2_{21} & a2_{22} \end{bmatrix} \tag{2}$$

Similarly,

$$B_1 = \begin{bmatrix} b1_{11} & b1_{12} & \cdots & b1_{130} \\ b1_{21} & b1_{22} & \cdots & b1_{230} \end{bmatrix} \tag{3}$$

and

$$B_2 = \begin{bmatrix} b2_{11} & b2_{12} & \cdots & b2_{130} \\ b2_{21} & b2_{22} & \cdots & b2_{230} \end{bmatrix} \tag{4}$$

Table 3: Malware Samples in CSV for Naïve K-means Method

malware	family	$a11$	$\cdots$	$a22$	$b11$	$\cdots$	$b130$	$b21$	$\cdots$	$b230$
file1	adload	$a1_{11}$	$\cdots$	$a1_{22}$	$b1_{11}$	$\cdots$	$b1_{130}$	$b1_{21}$	$\cdots$	$b1_{230}$
file2	adload	$a2_{11}$	$\cdots$	$a2_{22}$	$b2_{11}$	$\cdots$	$b2_{130}$	$b2_{21}$	$\cdots$	$b2_{230}$

Then the CSV file is given as the Table 3. The column $a11$ corresponds to the element in first row and first column of any matrix A. Similarly, $a22$ denotes element in second row and second column of the matrix A. Consequently, $b11$ and $b130$ are elements of any matrix B in first row, and first and thirteeth column respectively; $b21$ and $b230$ are elements in that B matrix in the second row, and first and thirteeth column respectively. We did this for all HMMs that were obtained from the malware opcodes. All the columns of B matrix in this CSV are given as feature vector input for the naïve k-means clustering algorithm with Euclidean distance metric as given in Chapter 2. The process was repeated for values of N from 2 to 4, and for all values of k from 2 to 15. We began by finding the optimal k using the elbow method discussed in Chapter 2 and did not expect to get a sharp elbow as our k-means is naïve. Figure 3 shows the graph obtained for our elbow method.

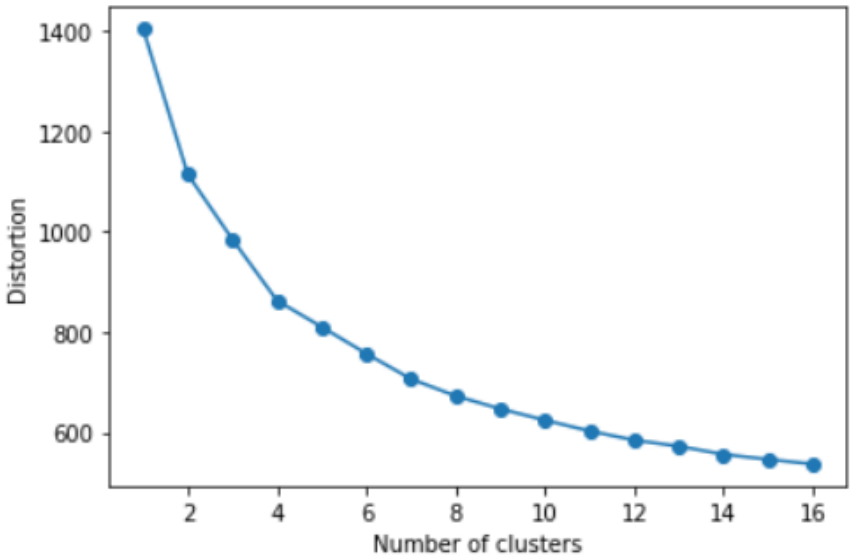

Figure 3: Elbow graph for the Naïve k-means

4.4.2 The K-medoids Method with Matrix Euclidean Distance

In another experiment, we compare the HMMs directly without converting the B matrices to CSV features, again using Euclidean distance as given in Chapter 2. However, we have used k-medoids clustering algorithm, which is a slight variation of the k-means algorithm, as discussed in Chapter 2. This method takes into account the flipping of states when computing HMMs.

Consider the two malware samples, $\lambda_1 = (A_1, B_1, \pi_1)$ and $\lambda_2 = (A_2, B_2, \pi_2)$, from above. Again, $N = 2$ and $M = 30$, and we do not consider this method for values of N other than 2. This time, consider the B matrix row-wise. We can then give the vector

$$b1 = (b11, b12)$$

where $b11$ is first row of B_1 and $b12$ is the second row and $b1$ is a vector of length $2M$. Similarly,

$$b2 = (b21, b22)$$

is a vector where $b21$ is first row of B_2 and $b22$ is the second row. We then computed

distance between these 2 HMMs as $d(b1, b2)$. Another vector

$$b3 = (b22, b21)$$

accounts for the flipped states of HMM. We give the distance between them as $d(b1, b3)$. The minimum of these distances is considered as the distance between the two models.

The next step is to compute a distance correlation matrix to use in k-medoids clustering algorithm. For this example, consider two more malware samples in the system, with HMMs as $\lambda_3 = (A_3, B_3, \pi_3)$ and $\lambda_4 = (A_4, B_4, \pi_4)$. The B matrices can be given as follows:

$$B_3 = \begin{bmatrix} b3_{11} & b3_{12} & \cdots & b3_{130} \\ b3_{21} & b3_{22} & \cdots & b3_{230} \end{bmatrix} \tag{5}$$

and

$$B_4 = \begin{bmatrix} b4_{11} & b4_{12} & \cdots & b4_{130} \\ b4_{21} & b4_{22} & \cdots & b4_{230} \end{bmatrix} \tag{6}$$

Table 4: Distance Correlation Matrix computed for K-medoids Method

B	B_1	B_2	B_3	B_4
B_1	0	$d(B_1, B_2)$	$d(B_1, B_3)$	$d(B_1, B_4)$
B_2	$d(B_2, B_1)$	0	$d(B_2, B_3)$	$d(B_2, B_4)$
B_3	$d(B_3, B_1)$	$d(B_3, B_2)$	0	$d(B_3, B_4)$
B_4	$d(B_4, B_1)$	$d(B_4, B_2)$	$d(B_4, B_3)$	0

Then the distance correlation matrix can be given as Table 4. Each distance is given by following the method above, considering flipping of the states of HMM. This distance correlation matrix can be given as input data to the k-medoids clustering algorithm, which is discussed in Chapter 2. The experiment was repeated for all values of k from 2 to 15. K different medoids are chosen randomly from the samples in the dataset for each of these k clusters.

4.4.3 Cluster Entropy

To check how good our clusters are, we calculated the entropies of each cluster in each of the experiments. Entropy is a general measure of the randomness of a

cluster [16]. Lower entropy within a cluster and higher entropy between clusters are indicative of good clustering.

Consider $X_1, X_2, \ldots, X_n$ as data points in a dataset, and let $C_1, C_2, \ldots, C_K$ be the clusters. Let M_j be the number of elements in cluster C_j. Then we can define M_{ij} as the number of elements of type i in cluster C_j, with $i = 1, 2, \ldots, l$ where l is the number of different families or types of data in the dataset. We can give the probability of data of type i in cluster j as $p_{ij} = M_{ij}/M_j$. Hence the entropy of the cluster C_j is given as

$$E_j = -\sum_{i=1}^{l} p_{ij} \log(p_{ij})$$

Then the weighted intra-cluster entropy is given as

$$E = \frac{1}{n} \sum_{i=1}^{K} M_j E_j$$

4.5 Discussion on Results

We obtained results as graphs for both the experiments mentioned above and have discussed them in detail in the following sections.

4.5.1 K-means

As we know from Table 2, the families Vbinject, Winwebsec [10], Vobfus, Ceeinject and Zegost are most dominant in the clustering. This is best visible in Figure 4, which depicts clustering with $k = 8$ for 4 hidden states of HMM.

Clearly, cluster 3 contains all the samples from the Vobfus family. Winwebsec family is dominant in cluster 0, but also constitutes clusters 2 and 7. Even though Vbinject family has most samples, they fail to be categorized in a single cluster. Most of Ceeinject family is found in cluster 2, with traces in clusters 5 and 7. Similarly, Zegost family is distributed between clusters 0, 2, 5 and 7.

The other families in the dataset are Zbot, Adload, Bho, Bifrose, Buzus, Delfinject, Dorkbot, Hotbar, Hupigon, Ircbot, Obfuscator and Rbot. Their contribution to the

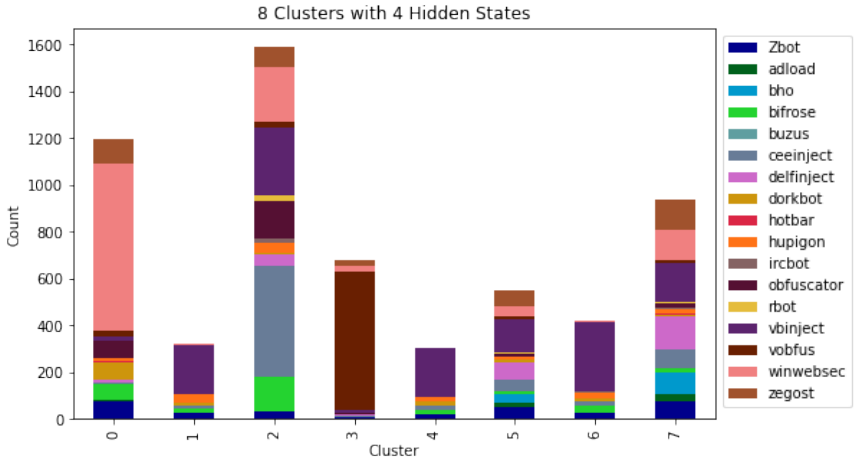

Figure 4: Stacked column chart for 8 clusters and 4 HMM hidden states with k-means

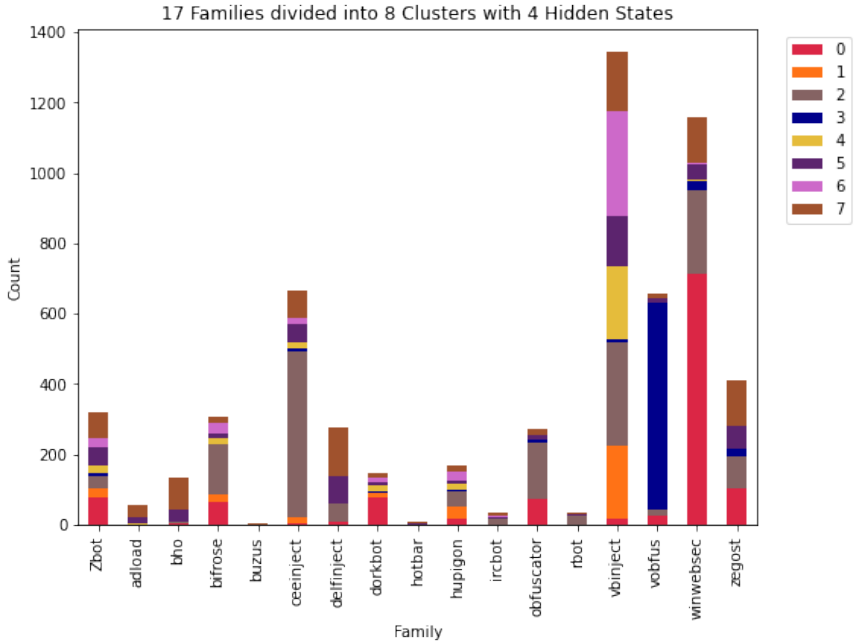

Figure 5: Stacked column chart for 17 families and 4 HMM hidden states with k-means

dataset is insignificant and is indicated accordingly in Figure 4.

From Figure 4, we also obtained the graph in Figure 5. Clearly, Winwebsec family has most malware samples from Cluster 0 and Vobfus family has malware samples from Cluster 3. Ceeinject family derives most malware samples from Cluster 2. Vbinject, despite being the dominant family, has malware samples divided roughly

equally divided in all clusters.

Table 5: Clustering Entropy for k-means Method with all k's and $N = 2$

Cluster	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	E
$k = 2$	2.292	**1.665**	-	-	-	-	-	-	-	-	-	-	-	-	-	1.841
$k = 3$	2.255	1.532	**0.947**	-	-	-	-	-	-	-	-	-	-	-	-	1.558
$k = 4$	**0.846**	1.525	1.909	2.212	-	-	-	-	-	-	-	-	-	-	-	1.760
$k = 5$	**0.810**	2.196	2.113	1.159	1.905	-	-	-	-	-	-	-	-	-	-	1.724
$k = 6$	2.136	1.544	1.939	1.139	2.189	**0.683**	-	-	-	-	-	-	-	-	-	1.711
$k = 7$	1.564	1.545	1.155	2.210	1.809	**0.646**	1.805	-	-	-	-	-	-	-	-	1.626
$k = 8$	1.813	2.197	**0.664**	1.657	1.612	1.147	2.080	1.440	-	-	-	-	-	-	-	1.555
$k = 9$	1.913	2.204	1.614	1.913	**0.641**	1.152	1.581	1.864	1.390	-	-	-	-	-	-	1.524
$k = 10$	1.166	1.953	1.686	1.516	**0.639**	1.328	2.142	2.291	1.327	1.716	-	-	-	-	-	1.667
$k = 11$	1.166	2.000	1.584	2.143	1.343	1.254	1.685	1.511	1.705	**0.352**	2.279	-	-	-	-	1.322
$k = 12$	2.267	1.309	2.294	1.497	1.386	1.671	1.101	1.717	1.509	2.072	1.964	**0.606**	-	-	-	1.585
$k = 13$	2.243	2.082	1.681	1.343	1.969	1.530	1.529	1.082	**0.376**	1.084	1.425	2.221	1.692	-	-	1.731
$k = 14$	1.601	1.471	2.291	1.481	1.125	2.293	0.393	**0.302**	1.710	1.452	1.167	1.975	1.701	0.893	-	1.450
$k = 15$	1.767	2.218	**0.362**	2.138	2.338	1.432	2.183	1.863	1.552	1.674	1.050	1.567	0.910	1.344	0.433	1.514

Table 6: Clustering Entropy for k-means Method with all k's and $N = 3$

Cluster	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	E
$k = 2$	**1.609**	2.306	-	-	-	-	-	-	-	-	-	-	-	-	-	2.110
$k = 3$	**0.818**	1.430	2.288	-	-	-	-	-	-	-	-	-	-	-	-	1.482
$k = 4$	2.233	**0.737**	1.980	1.500	-	-	-	-	-	-	-	-	-	-	-	1.782
$k = 5$	1.721	1.941	1.128	**0.677**	2.288	-	-	-	-	-	-	-	-	-	-	1.376
$k = 6$	**0.682**	2.263	1.355	1.562	1.237	1.895	-	-	-	-	-	-	-	-	-	1.273
$k = 7$	**0.647**	1.880	2.426	1.279	1.262	2.216	1.287	-	-	-	-	-	-	-	-	1.384
$k = 8$	2.021	2.236	1.233	1.775	**0.850**	1.614	1.357	0.876	-	-	-	-	-	-	-	1.531
$k = 9$	0.978	2.005	1.800	**0.605**	2.147	1.318	2.323	2.382	1.389	-	-	-	-	-	-	1.748
$k = 10$	2.348	2.350	0.846	1.918	1.242	**0.757**	1.163	0.993	1.690	2.312	-	-	-	-	-	1.717
$k = 11$	1.825	2.145	2.324	**0.744**	1.168	1.292	1.027	1.895	2.356	0.762	1.669	-	-	-	-	1.422
$k = 12$	2.007	2.333	1.240	**0.763**	2.305	1.363	1.095	2.315	2.010	0.992	1.143	1.057	-	-	-	1.591
$k = 13$	1.383	2.286	1.287	1.631	1.027	2.007	0.966	2.000	1.110	2.328	**0.751**	0.806	2.330	-	-	1.609
$k = 14$	**0.810**	2.322	1.559	1.024	1.195	1.770	2.350	1.274	1.912	1.133	2.122	1.011	0.946	2.249	-	1.397
$k = 15$	1.982	1.235	0.941	2.285	2.332	1.292	1.029	1.135	1.399	1.232	2.313	1.998	**0.319**	2.118	1.137	1.414

Table 7: Clustering Entropy for k-means Method with all k's and $N = 4$

Cluster	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	E
$k = 2$	2.297	**1.607**	-	-	-	-	-	-	-	-	-	-	-	-	-	1.801
$k = 3$	**0.786**	2.281	1.363	-	-	-	-	-	-	-	-	-	-	-	-	1.792
$k = 4$	1.926	1.223	**0.682**	2.297	-	-	-	-	-	-	-	-	-	-	-	1.201
$k = 5$	1.256	1.920	1.214	2.299	**0.662**	-	-	-	-	-	-	-	-	-	-	1.308
$k = 6$	1.269	2.130	2.273	1.272	**0.632**	1.808	-	-	-	-	-	-	-	-	-	1.388
$k = 7$	2.182	1.262	2.093	**0.622**	1.220	1.799	2.255	-	-	-	-	-	-	-	-	1.526
$k = 8$	1.501	1.286	2.058	**0.633**	1.212	2.302	1.112	2.321	-	-	-	-	-	-	-	1.779
$k = 9$	2.232	**0.702**	2.276	1.318	1.768	1.211	2.049	1.175	0.967	-	-	-	-	-	-	1.495
$k = 10$	**0.627**	2.333	1.158	1.490	2.344	2.302	2.050	1.140	1.215	1.947	-	-	-	-	-	1.579
$k = 11$	1.147	2.306	2.143	**0.735**	1.719	0.882	2.088	1.176	1.262	1.245	2.260	-	-	-	-	1.447
$k = 12$	2.318	1.686	1.171	2.267	1.229	2.263	0.833	1.232	2.062	1.086	2.051	**0.691**	-	-	-	1.573
$k = 13$	0.864	1.098	2.014	1.915	**0.732**	2.304	1.143	2.286	1.218	1.419	1.174	2.327	2.339	-	-	1.579
$k = 14$	1.205	2.294	**0.431**	1.442	2.309	1.995	1.127	1.614	2.272	1.235	1.085	0.937	1.796	1.911	-	1.604
$k = 15$	2.299	2.264	1.848	1.155	2.269	**0.646**	0.826	1.254	1.066	2.041	1.651	2.012	1.222	1.074	1.013	1.507

We also computed the entropies for all the clusters, for all values of k from 2 to 15, and all values of N from 2 to 4. Table 5 contains the entropies for $N = 2$ while

Table 6 for $N = 3$. In Table 7, which shows the entropies for $N = 4$, Cluster 3 has an entropy of 0.633 for $k = 8$, and from Figure 4 we know Cluster 3 shows the most clear cluster of samples from Vobfus family. Whereas, Cluster 7 has the highest entropy, meaning it is the most impure cluster. This is evident in Figure 4. All the entropies in Tables 5, 6 and 7 can be verified as corresponding to the graphs in Appendix A. The best result for each experiment from $k = 2$ to $k = 15$ is written in bold, while the best result for each cluster is placed in a box.

K-means clustering algorithm has a major drawback of choosing the number of clusters k before using the algorithm. We have performed experiments for $k = 2$ to $k = 15$ for HMM hidden states N from 2 to 4 and discussed the best case at $k = 8$. All other results can be found in Appendix A and Appendix C.

4.5.2 K-medoids

As we also experimented with k-medoids algorithm, we also discuss the results for it here. Figure 6 depicts clustering for $k = 8$ and 2 hidden states of HMM.

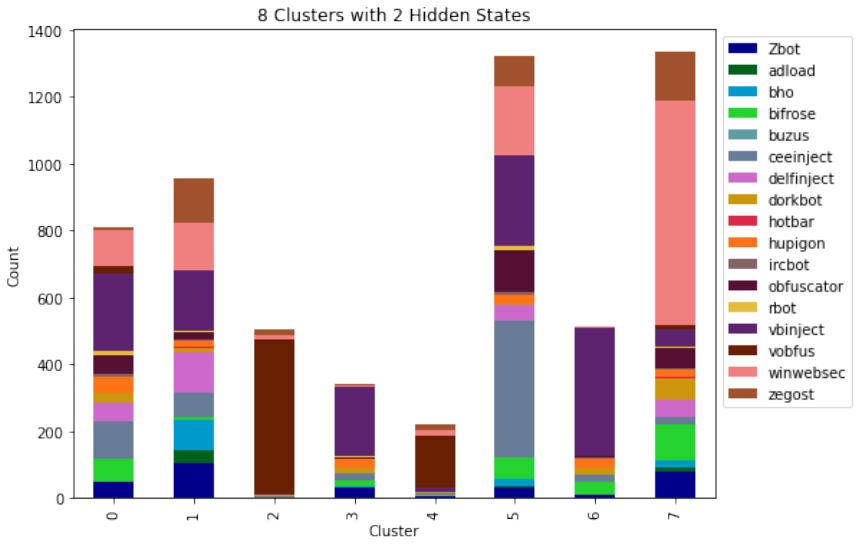

Figure 6: Stacked column chart for 8 clusters and 2 HMM hidden states with k-medoids

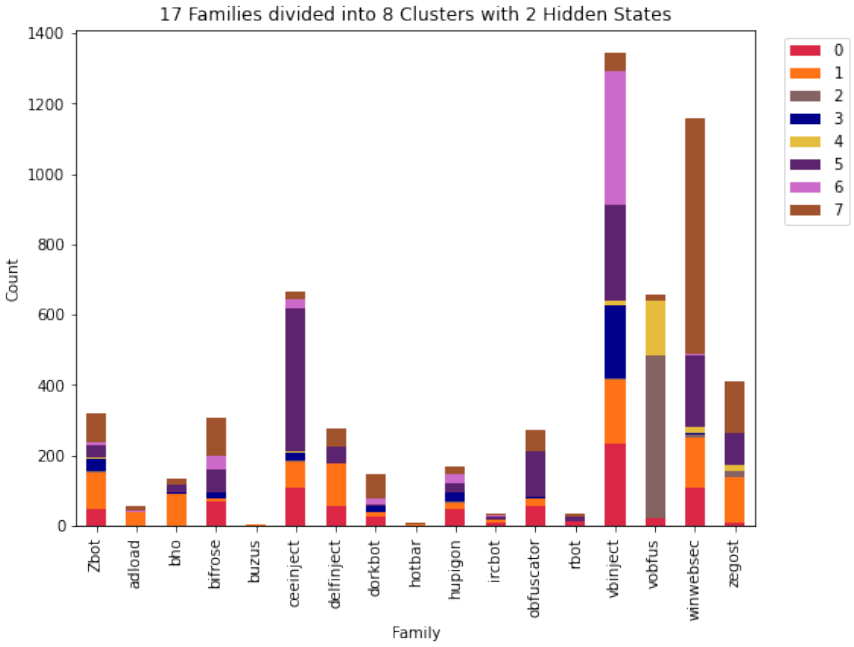

Figure 7: Stacked column chart for 17 families and 2 HMM hidden states with k-medoids

The results are much more divided in this experiment. No single cluster contains all the files from one family. Vobfus family is divided into Cluster 2 and 4. Vbinject family is also divided into all clusters except 2 and 4. Winwebsec family is present in all clusters, but is dominant in Cluster 7. Ceeinject family is dominant in Cluster 5 but is also spread into other clusters.

We can confirm these deductions by obtaining the graph in Figure 7. As pointed out earlier, Winwebsec family is derived from Cluster 7, while Vofbus family has most samples from Cluster 2.

We again computed the entropies for all the clusters, for all values of k from 2 to 15. These are given in Table 8 below. Cluster 2 has an entropy of 0.664 and in Figure 6 cluster 2 shows the most clear cluster of samples from Vobfus family. Cluster 1 has the highest entropy indicating the most impure cluster. We can verify all the entropies in Table 8 as corresponding to the graphs in Appendix B. The best result

Table 8: Clustering Entropy for k-medoids Method with all k's and $N = 2$

Cluster	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	E
$k = 2$	**1.860**	2.315	-	-	-	-	-	-	-	-	-	-	-	-	-	2.187
$k = 3$	1.709	2.223	**1.438**	-	-	-	-	-	-	-	-	-	-	-	-	1.959
$k = 4$	2.195	**0.749**	2.217	1.367	-	-	-	-	-	-	-	-	-	-	-	1.889
$k = 5$	1.898	1.611	2.203	1.061	**0.753**	-	-	-	-	-	-	-	-	-	-	1.915
$k = 6$	2.290	**0.516**	1.543	1.971	1.331	1.503	-	-	-	-	-	-	-	-	-	1.800
$k = 7$	1.996	1.862	1.202	1.540	2.239	1.639	**0.502**	-	-	-	-	-	-	-	-	1.852
$k = 8$	2.210	2.253	**0.441**	1.431	1.162	2.002	1.055	1.846	-	-	-	-	-	-	-	1.760
$k = 9$	1.889	1.186	1.431	1.541	2.257	**1.039**	1.055	1.852	1.995	-	-	-	-	-	-	1.794
$k = 10$	2.195	1.832	0.888	**0.282**	1.508	1.057	1.431	1.929	1.104	2.239	-	-	-	-	-	1.732
$k = 11$	2.347	1.518	**0.282**	1.346	1.960	1.896	2.205	0.888	1.036	1.925	1.085	-	-	-	-	1.732
$k = 12$	1.879	1.060	1.954	1.494	1.764	2.265	1.489	1.508	**0.474**	0.888	1.104	1.813	-	-	-	1.696
$k = 13$	2.090	2.343	1.426	1.109	1.928	0.986	0.888	1.607	0.373	1.528	1.892	1.887	**0.259**	-	-	1.651
$k = 14$	2.422	1.064	1.636	**0.109**	1.813	0.685	0.517	0.282	1.897	1.550	1.912	1.494	1.104	1.483	-	1.629
$k = 15$	2.322	1.801	1.413	1.585	0.847	1.876	1.060	1.031	1.752	**0.164**	0.995	**0.109**	1.783	0.663	2.193	1.663

for each experiment from $k = 2$ to $k = 15$ is written in bold, while the best result for each cluster is placed in a box.

K-medoids faces the issue of choosing number of clusters k, similar to k-means. We have performed experiments for $k = 2$ to $k = 15$ for HMM hidden states $N = 2$ and discussed the best case at $k = 8$. All other results can be found in Appendix B.

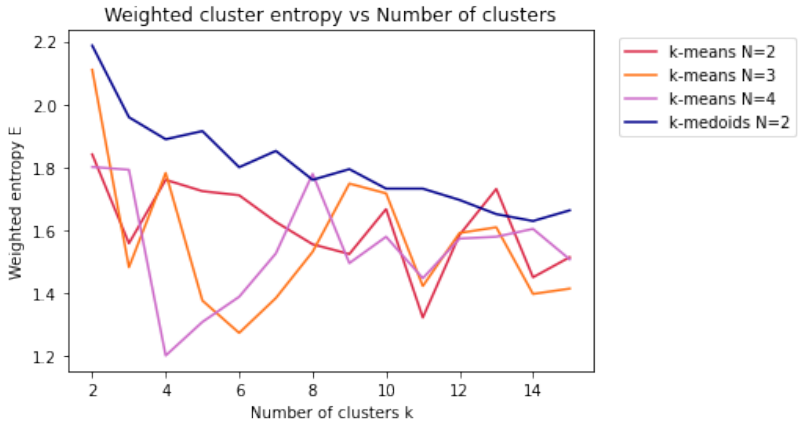

Figure 8: Comparison of Weighted Entropies as a function of k

We also compared the results of all our experiments by comparing the weighted intra-cluster entropies for all values of k from 2 to 15, and displayed the result as a graph given the Figure 8. It shows different trends for each experiment. All the naïve k-means experiments have fluctuating weighted entropies as the number of clusters change. K-means with $N = 2$ has a dip in the weighted entropy for $k = 11$, while for

$N = 3$ the dip is observed at $k = 6$. For $N = 4$, the weighted entropy is lowest at $k = 4$. However, we do not find spikes in the k-medoids method, although there is a downward trend in the entropy with increasing number of clusters.

CHAPTER 5

Conclusion and Future Work

In this project, we proposed and evaluated clustering algorithms for classifying about 5997 malware opcode samples. Instead of scoring the samples using hidden Markov models, we only computed the models on every malware sample. We then used the B matrices of each model as the difference between models and used it to calculate distances. We used these B matrices in two experiments. The first experiment converted the B matrices to a dataframe and directly used with a k-means clustering algorithm. The second experiment used custom distance calculation between B matrices of models and used it with a k-medoids clustering algorithm. We performed experiments for varying numbers of clusters from $k = 2$ to $k = 15$. We then plotted stacked column charts for all the cases. The case $k = 8$ looks to be the best in both these experiments. The distinction between families was clearly visible in the clusters in this case. The results showed that HMMs are effective in classifying malware automatically.

We calculated the distance for the clustering algorithms using the Euclidean distance formula in this project. Other methods like Minkowski distance or cosine distance can be explored in future work. Another idea for future work would be to compute the distance between HMMs using the technique given in [14].

We used two types of centroid selection for clustering. We have computed the cluster centroids by calculating the mean of the respective clusters for experiments with the k-means method. For the k-medoids method, the centroids were randomly selected from within the data. Spherical k-means clustering, expectation-maximization (EM) clustering, and Density-based Spatial Clustering of Applications with Noise (DBSCAN) can also be explored as clustering options in the future.

Some of the malware samples are very different than others. We can filter these

out from the dataset for much accurate clustering in the future. Also, in the future, we can train the HMMs only for the dominant malware families in the dataset to see how the clusters are affected.

LIST OF REFERENCES

[1] A. Vasudevan, ''MalTRAK: Tracking and eliminating unknown malware,'' in *2008 Annual Computer Security Applications Conference (ACSAC)*, 2008, pp. 311--321.

[2] J. Aycock, *Computer viruses and malware*. Springer Science & Business Media, September 2006, vol. 22.

[3] M. Stamp, *Information Security: Principles and Practice*. Wiley, 2011.

[4] C. Annachhatre, T. H. Austin, and M. Stamp, ''Hidden Markov models for malware classification,'' *Journal of Computer Virology and Hacking Techniques*, vol. 11, no. 2, pp. 59--73, 2015. [Online]. Available: https://doi-org.libaccess.sjlibrary.org/10.1007/s11416-014-0215-x

[5] ''2021 cyber security statistics,'' 2021. [Online]. Available: https://purplesec.us/resources/cyber-security-statistics/

[6] ''Win32/zbot,'' 2017. [Online]. Available: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32%2FZbot

[7] ''Zeus (malware),'' 2010. [Online]. Available: https://en.wikipedia.org/wiki/Zeus_(malware)

[8] ''Backdoor computing attacks,'' 2021. [Online]. Available: https://www.malwarebytes.com/backdoor/

[9] ''What is spyware? and how to remove it,'' 2019. [Online]. Available: https://us.norton.com/internetsecurity-how-to-catch-spyware-before-it-snags-you.html

[10] ''Win32/winwebsec,'' 2017. [Online]. Available: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win32%2fWinwebsec

[11] H. El Merabet and A. Hajraoui, ''A survey of malware detection techniques based on machine learning,'' *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 1, pp. 366--373, 2019.

[12] I. Firdausi, A. Erwin, A. S. Nugroho, *et al.*, ''Analysis of machine learning techniques used in behavior-based malware detection,'' in *2010 Second Int. Conf. on Advances in Computing, Control, and Telecommunication Technologies*. IEEE, 2010, pp. 201--203.

[13] W. Hardy, L. Chen, S. Hou, Y. Ye, and X. Li, ''DL4MD: A deep learning framework for intelligent malware detection,'' in *Proceedings of the International Conference on Data Science (ICDATA)*, 2016, p. 61.

[14] L. R. Rabiner, ''A tutorial on hidden Markov models and selected applications in speech recognition,'' *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257--286, 1989.

[15] M. Stamp, ''A revealing introduction to hidden Markov models,'' pp. 26--56, 2004. [Online]. Available: http://www.cs.sjsu.edu/faculty/stamp/RUA/HMM.pdf

[16] M. Stamp, *Introduction to Machine Learning with Applications in Information Security*. CRC Press, 2017.

[17] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1988.

[18] J. LZP, ''What is the difference between hierarchical and partitional clustering?'' 2019. [Online]. Available: https://lzpdatascience.medium.com/what-is-the-difference-between-hierarchical-and-partitional-clustering-edc0d488c7c4

[19] A. Moore, ''K-means and Hierarchical Clustering,'' 2001. [Online]. Available: http://www.cs.cmu.edu/afs/cs/user/awm/web/tutorials/kmeans11.pdf

[20] P. J. Rousseeuw and L. Kaufman, ''Clustering by means of medoids,'' https://wis.kuleuven.be/stat/robust/papers/publications-1987/kaufmanrousseeuw-clusteringbymedoids-l1norm-1987.pdf, 1987.

[21] K. Mahendru, ''How to Determine the Optimal k for k-means?'' 2019. [Online]. Available: https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb

[22] S. Cesare and Y. Xiang, ''Classification of malware using structured control flow,'' in *Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing-Volume 107*, 2010, pp. 61--70.

[23] R. Canzanese, M. Kam, and S. Mancoridis, ''Toward an automatic, online behavioral malware classification system,'' in *2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*. IEEE, 2013, pp. 111--120.

[24] T. Singh, F. Di Troia, V. A. Corrado, T. H. Austin, and M. Stamp, ''Support Vector Machines and malware detection,'' *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 4, pp. 203--212, 2016.

[25] P. Thompson, ''Looking back: On relevance, probabilistic indexing and information retrieval,'' *Information processing & management*, vol. 44, no. 2, pp. 963--970, 2008.

[26] A. Lakhotia, A. Walenstein, C. Miles, and A. Singh, ‘‘VILO: a rapid learning nearest-neighbor classifier for malware triage,’’ *Journal of Computer Virology and Hacking Techniques*, vol. 9, no. 3, pp. 109--123, 2013.

APPENDIX A

Graphs for Naïve K-means Method

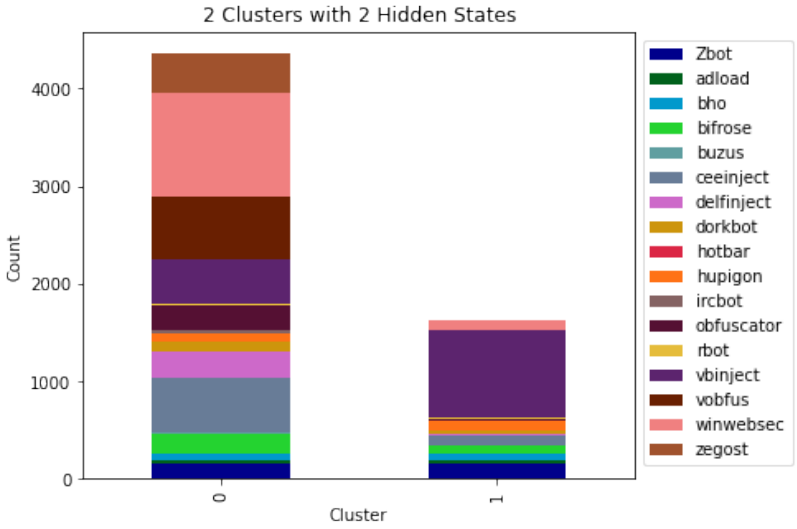

Figure A.9: Stacked column chart for 2 clusters and 2 hidden states in HMM

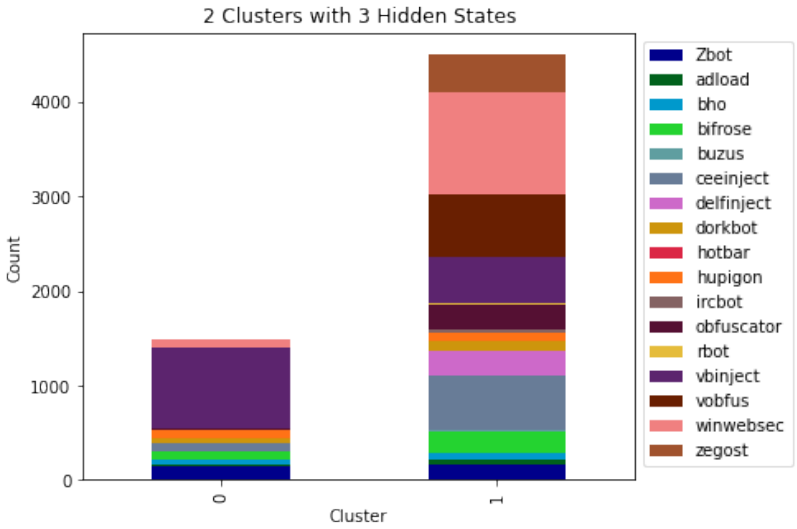

Figure A.10: Stacked column chart for 2 clusters and 3 hidden states in HMM

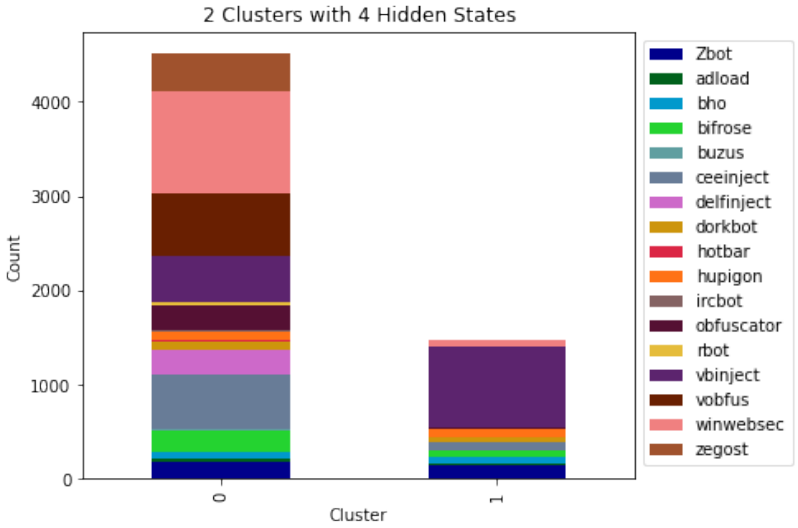

Figure A.11: Stacked column chart for 2 clusters and 4 hidden states in HMM

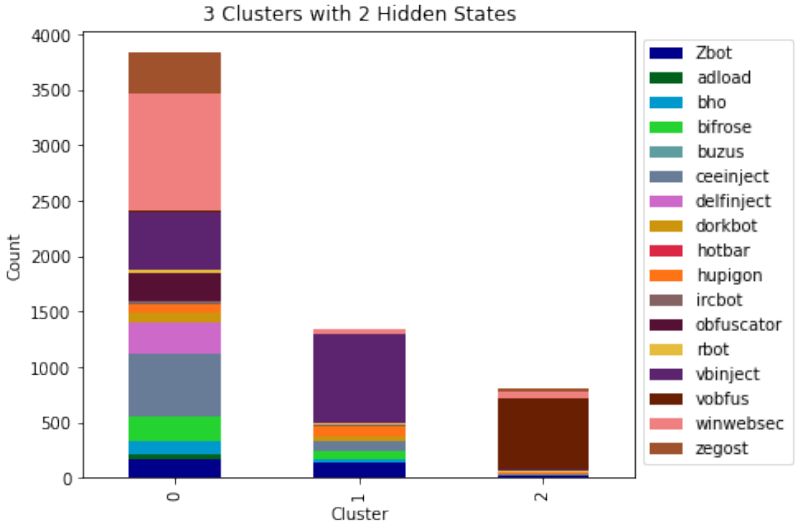

Figure A.12: Stacked column chart for 3 clusters and 2 hidden states in HMM

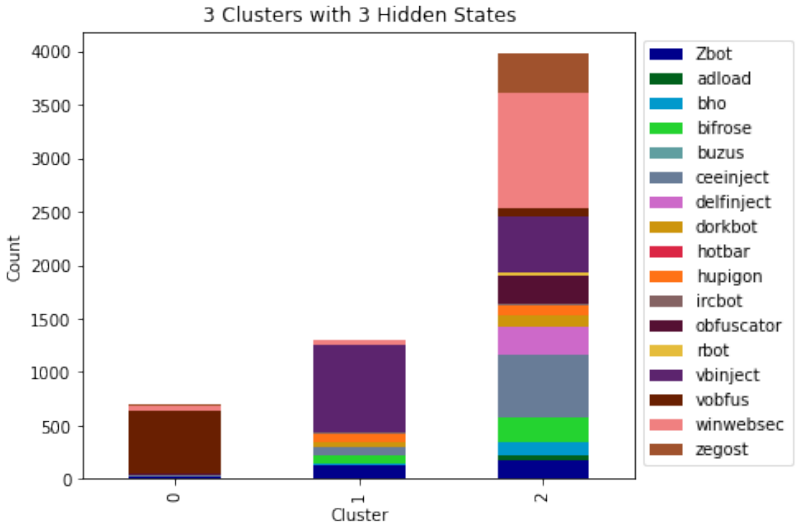

Figure A.13: Stacked column chart for 3 clusters and 3 hidden states in HMM

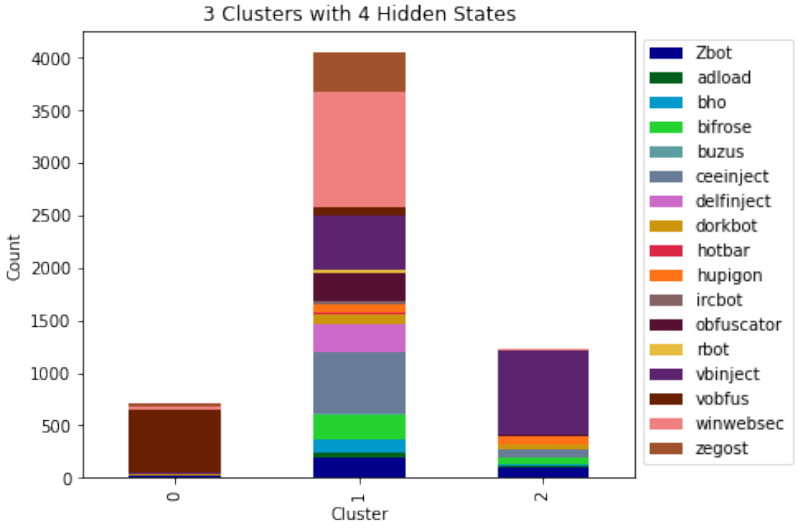

Figure A.14: Stacked column chart for 3 clusters and 4 hidden states in HMM

Figure A.15: Stacked column chart for 4 clusters and 2 hidden states in HMM

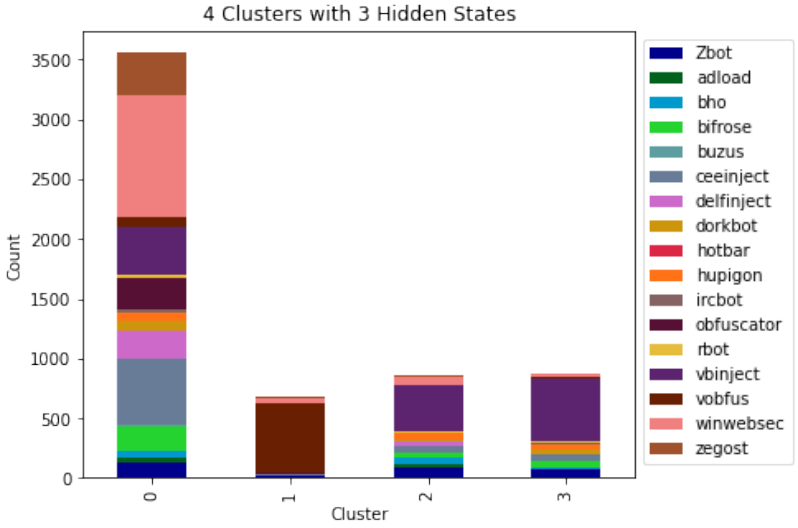

Figure A.16: Stacked column chart for 4 clusters and 3 hidden states in HMM

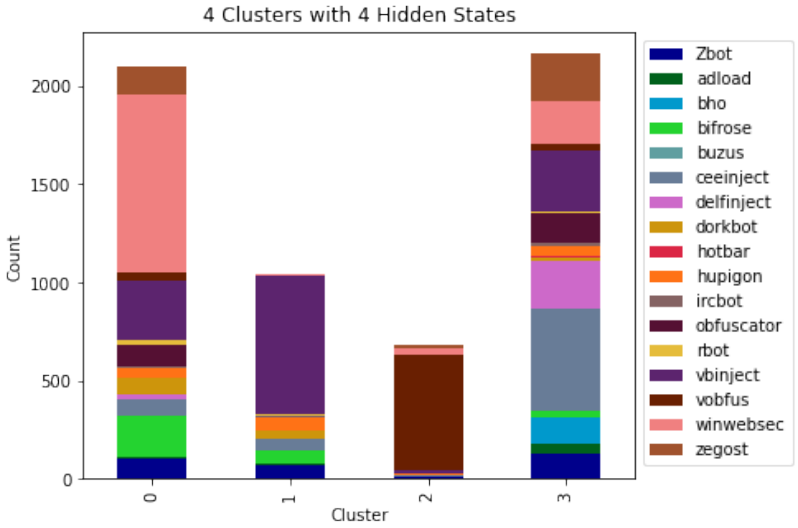

Figure A.17: Stacked column chart for 4 clusters and 4 hidden states in HMM

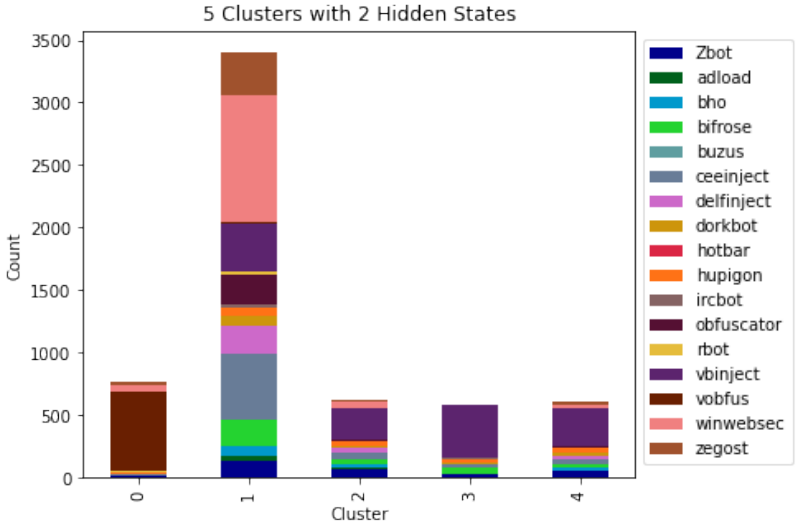

Figure A.18: Stacked column chart for 5 clusters and 2 hidden states in HMM

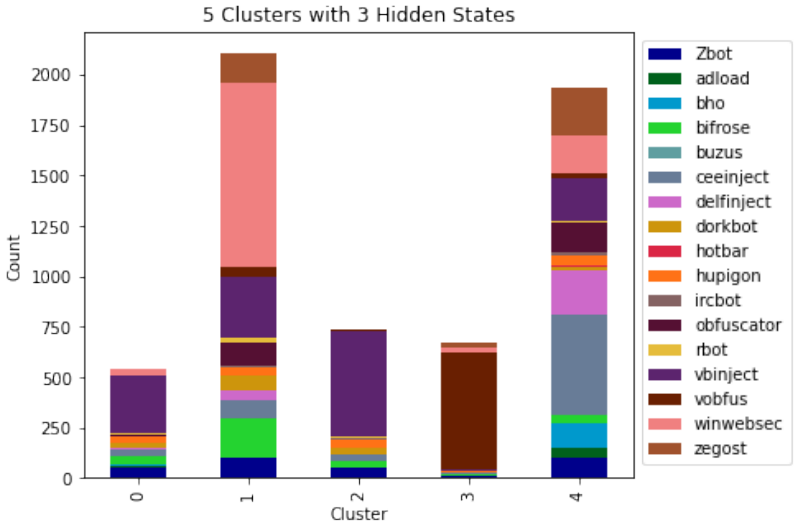

Figure A.19: Stacked column chart for 5 clusters and 3 hidden states in HMM

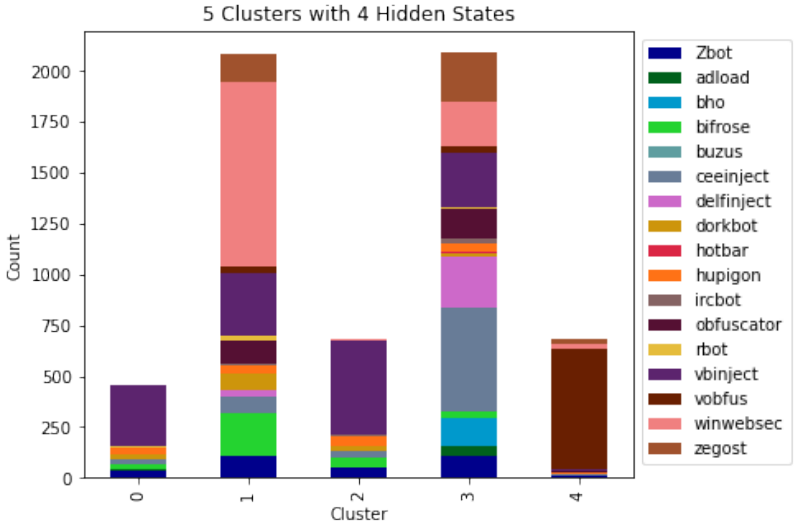

Figure A.20: Stacked column chart for 5 clusters and 4 hidden states in HMM

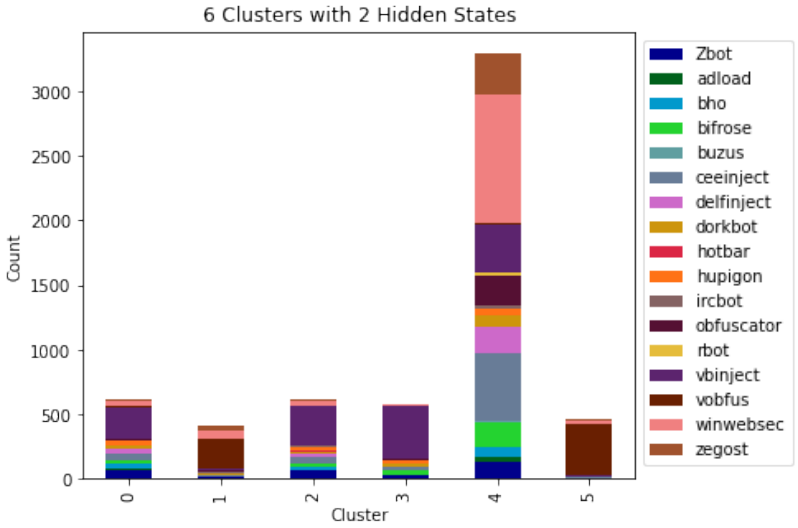

Figure A.21: Stacked column chart for 6 clusters and 2 hidden states in HMM

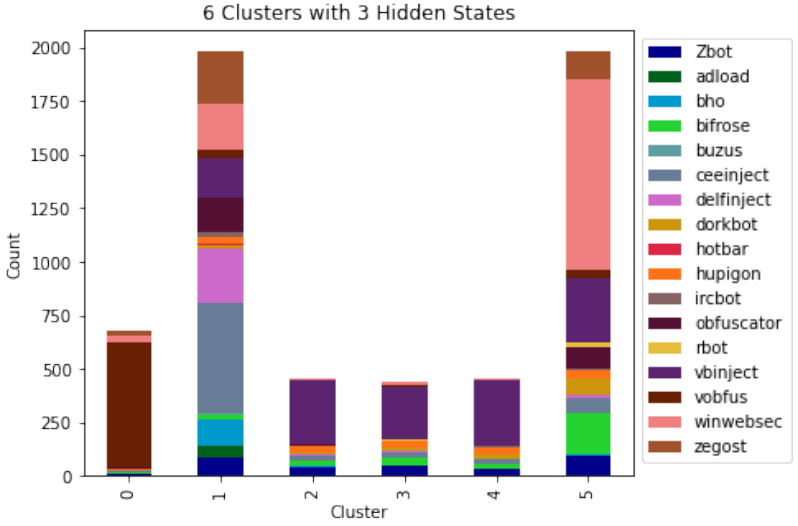

Figure A.22: Stacked column chart for 6 clusters and 3 hidden states in HMM

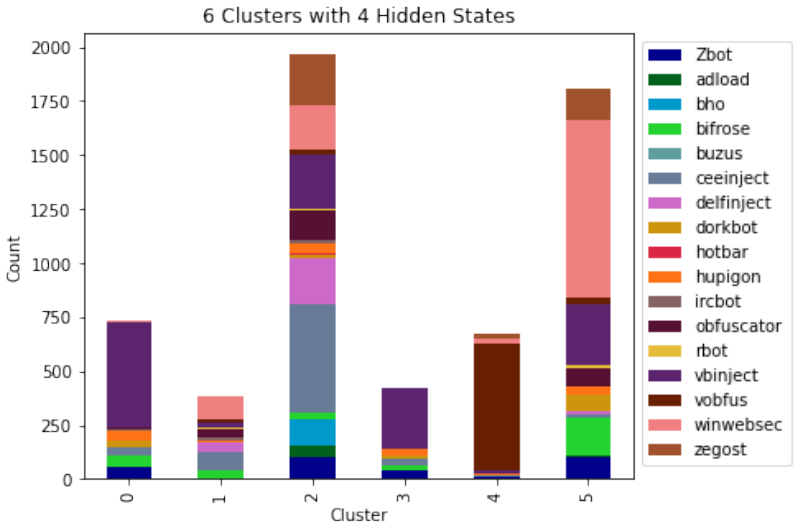

Figure A.23: Stacked column chart for 6 clusters and 4 hidden states in HMM

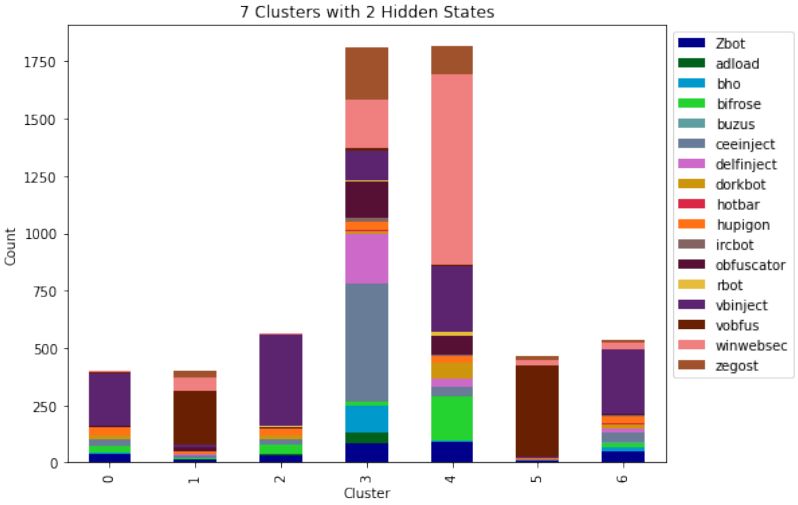

Figure A.24: Stacked column chart for 7 clusters and 2 hidden states in HMM

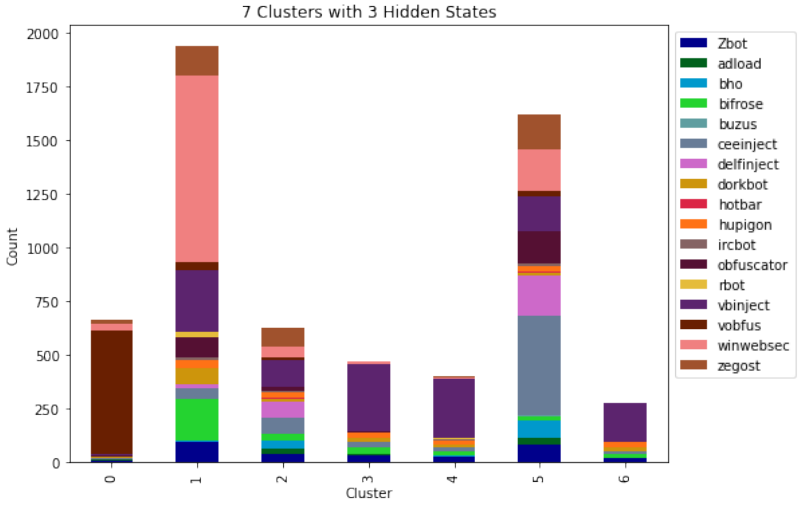

Figure A.25: Stacked column chart for 7 clusters and 3 hidden states in HMM

Figure A.26: Stacked column chart for 7 clusters and 4 hidden states in HMM

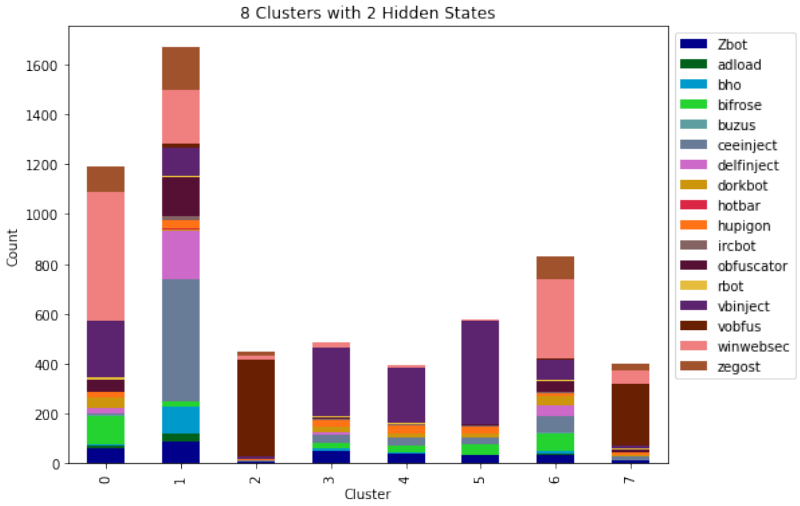

Figure A.27: Stacked column chart for 8 clusters and 2 hidden states in HMM

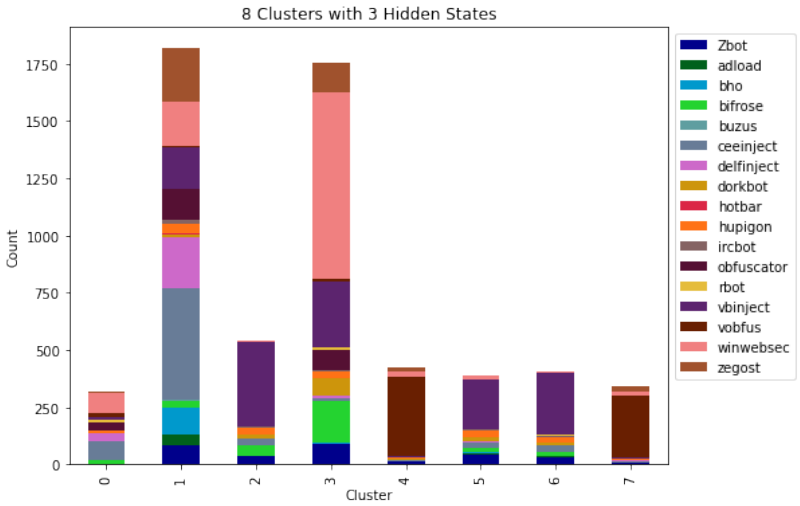

Figure A.28: Stacked column chart for 8 clusters and 3 hidden states in HMM

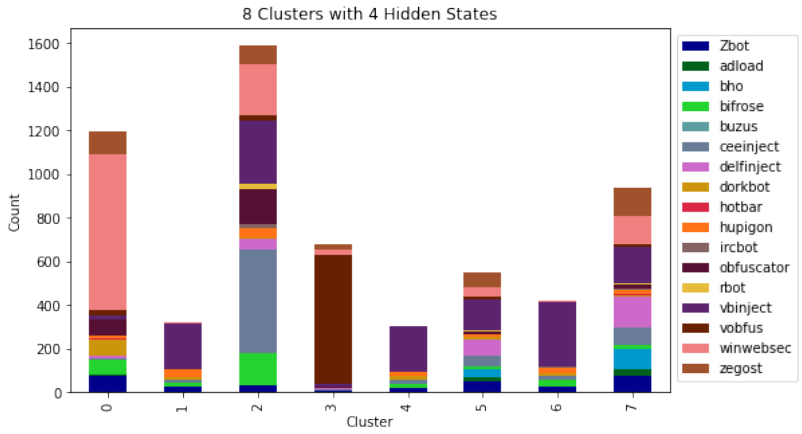

Figure A.29: Stacked column chart for 8 clusters and 4 hidden states in HMM

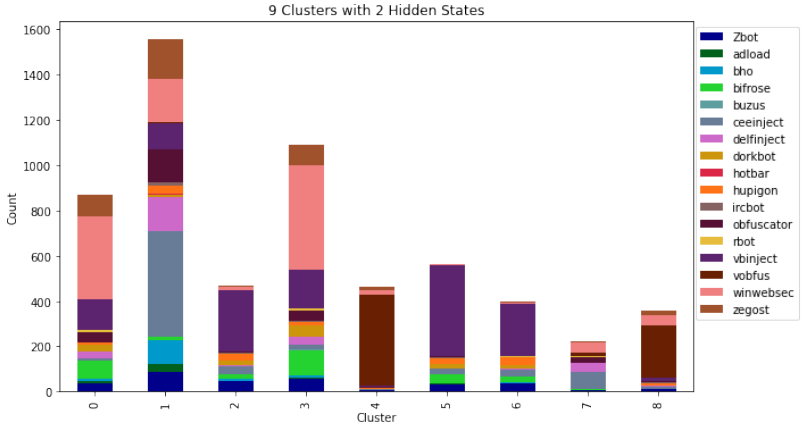

Figure A.30: Stacked column chart for 9 clusters and 2 hidden states in HMM

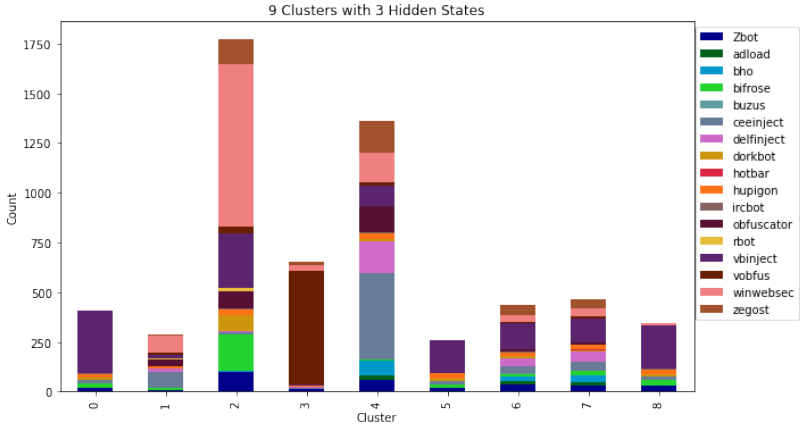

Figure A.31: Stacked column chart for 9 clusters and 3 hidden states in HMM

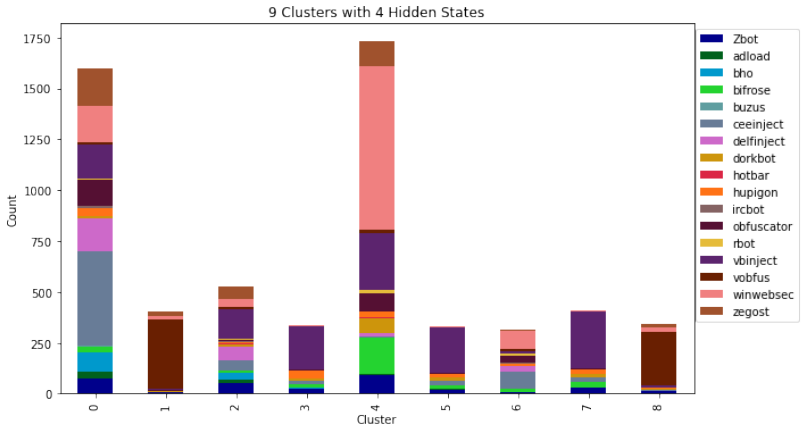

Figure A.32: Stacked column chart for 9 clusters and 4 hidden states in HMM

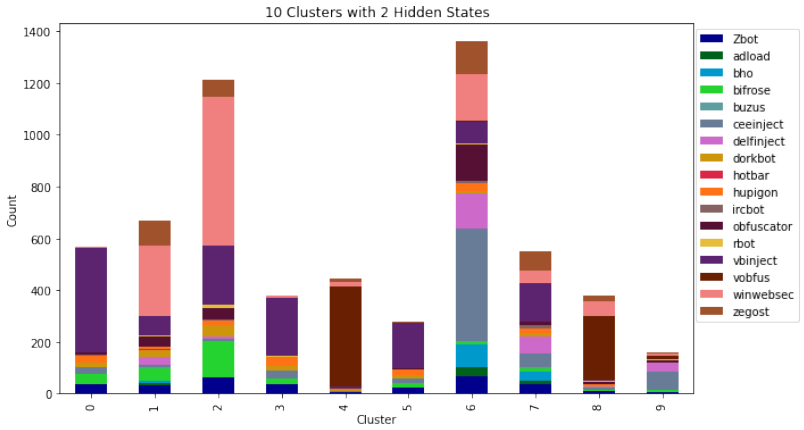

Figure A.33: Stacked column chart for 10 clusters and 2 hidden states in HMM

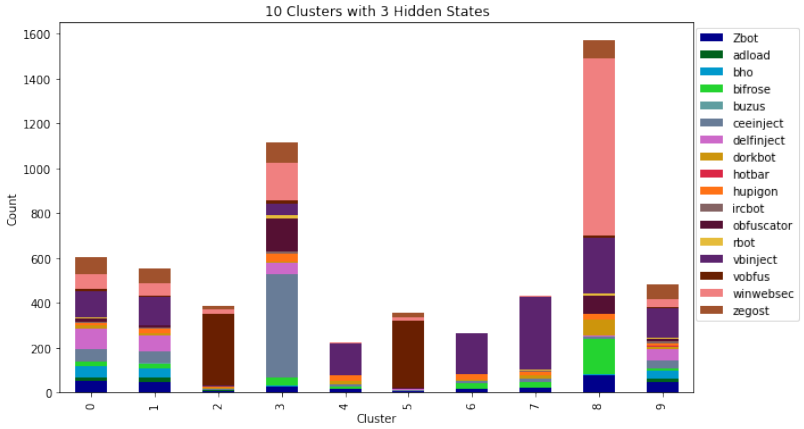

Figure A.34: Stacked column chart for 10 clusters and 3 hidden states in HMM

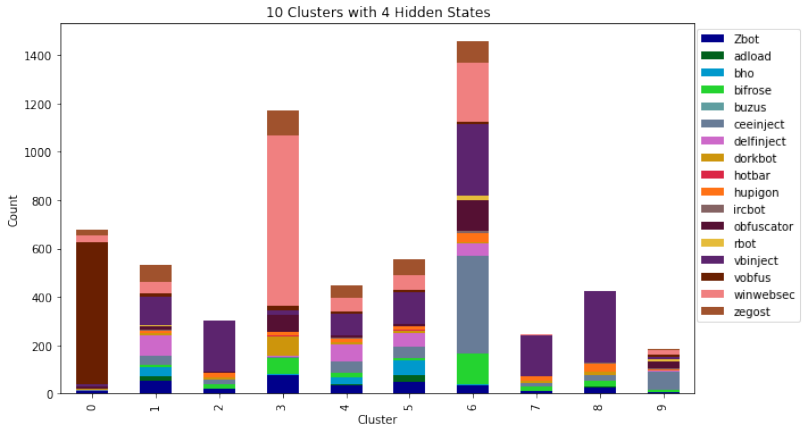

Figure A.35: Stacked column chart for 10 clusters and 4 hidden states in HMM

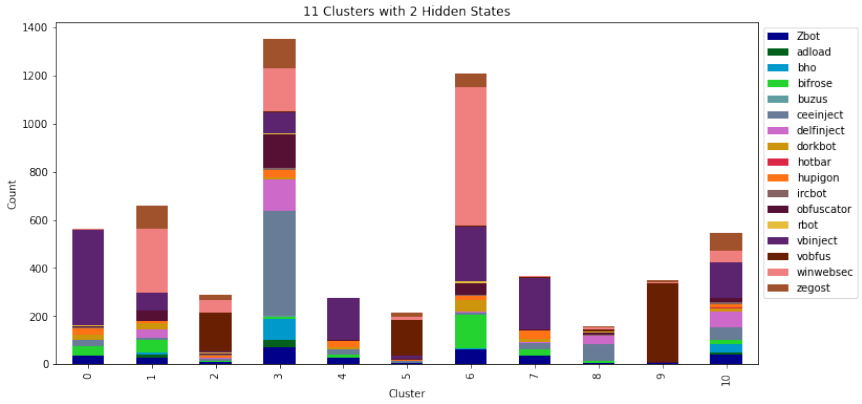

Figure A.36: Stacked column chart for 11 clusters and 2 hidden states in HMM

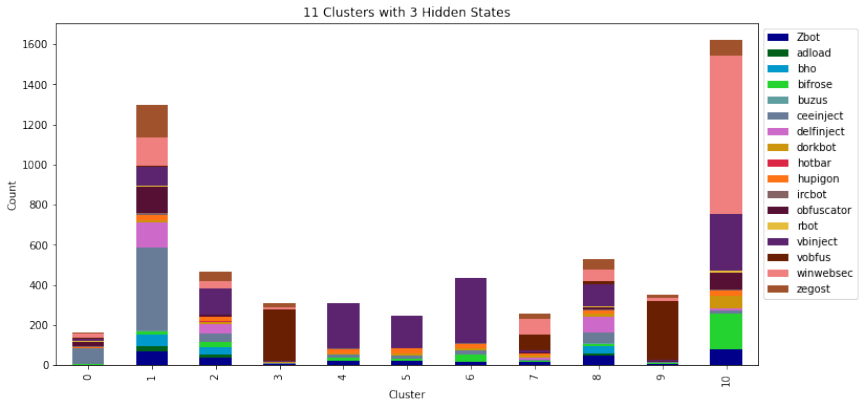

Figure A.37: Stacked column chart for 11 clusters and 3 hidden states in HMM

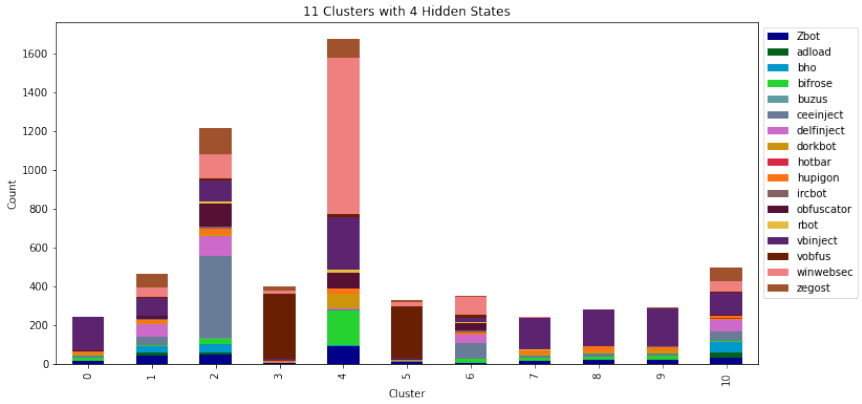

Figure A.38: Stacked column chart for 11 clusters and 4 hidden states in HMM

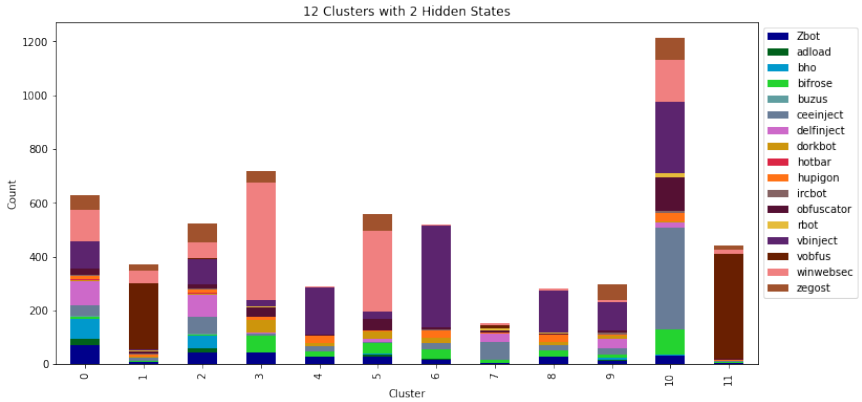

Figure A.39: Stacked column chart for 12 clusters and 2 hidden states in HMM

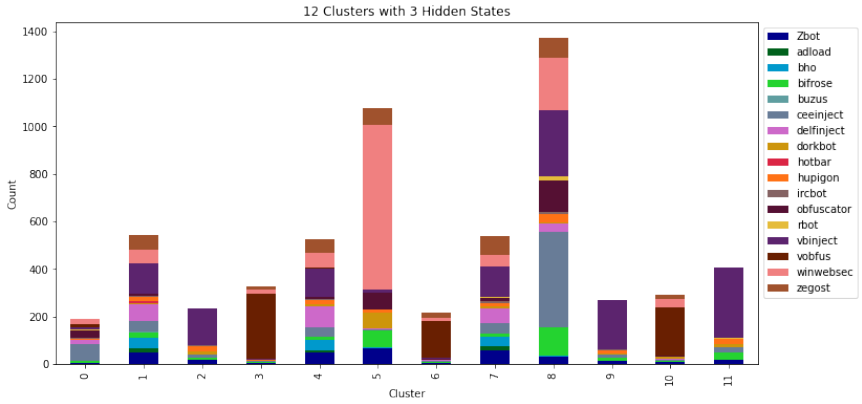

Figure A.40: Stacked column chart for 12 clusters and 3 hidden states in HMM

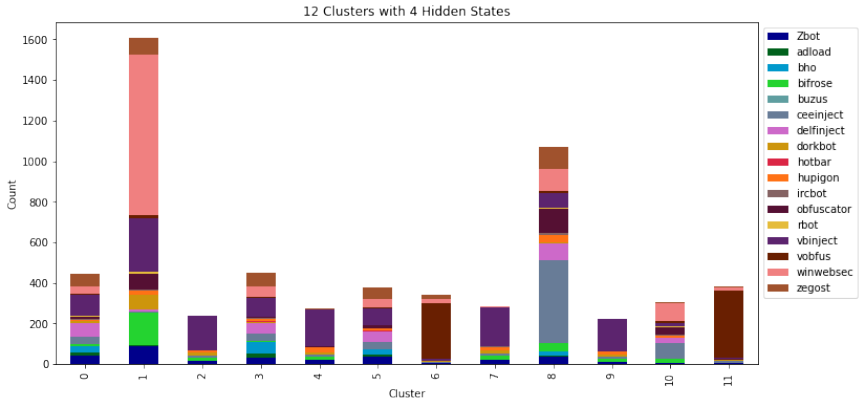

Figure A.41: Stacked column chart for 12 clusters and 4 hidden states in HMM

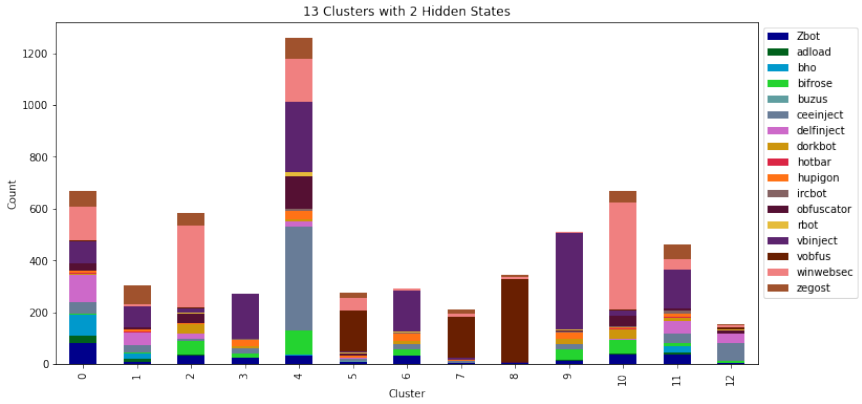

Figure A.42: Stacked column chart for 13 clusters and 2 hidden states in HMM

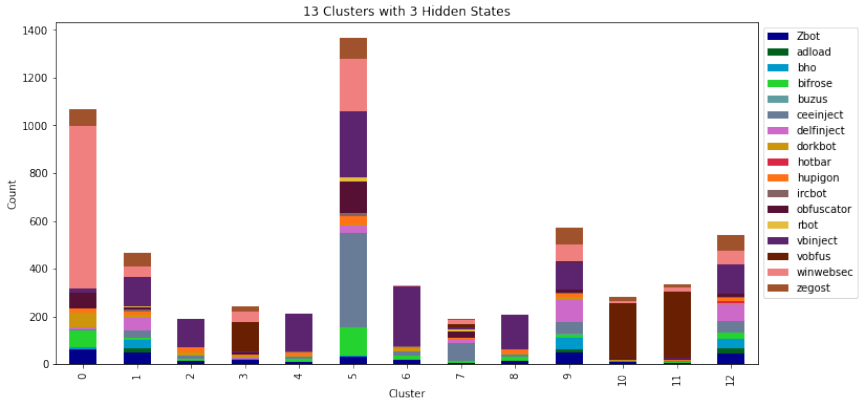

Figure A.43: Stacked column chart for 13 clusters and 3 hidden states in HMM

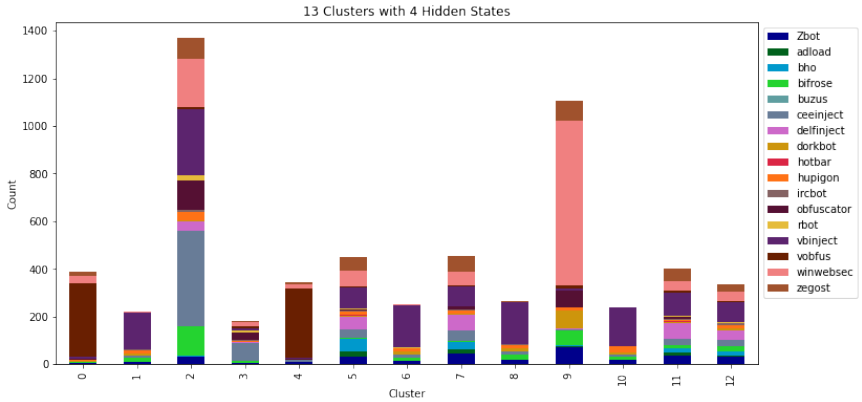

Figure A.44: Stacked column chart for 13 clusters and 4 hidden states in HMM

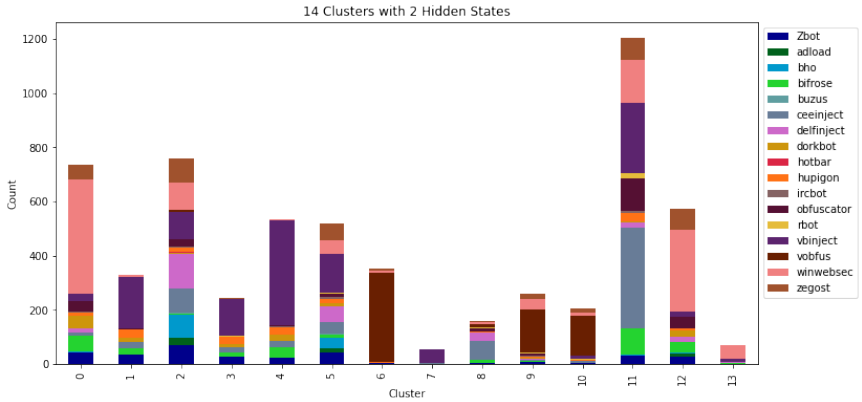

Figure A.45: Stacked column chart for 14 clusters and 2 hidden states in HMM

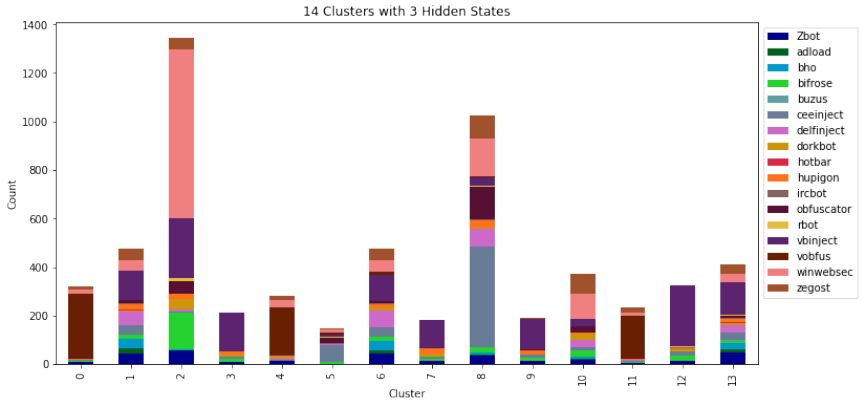

Figure A.46: Stacked column chart for 14 clusters and 3 hidden states in HMM

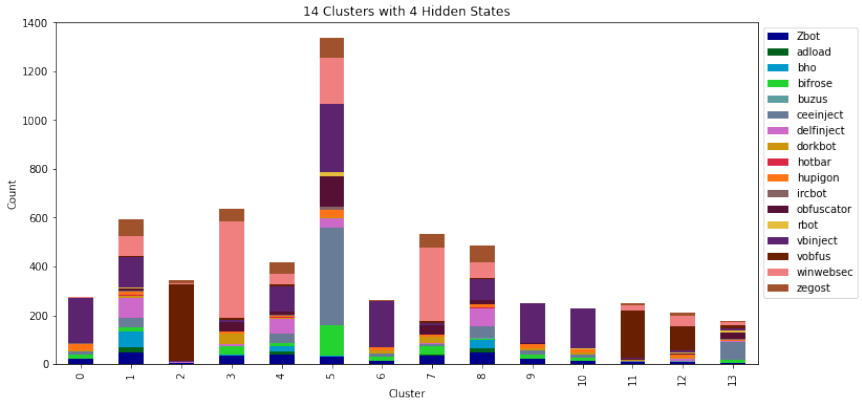

Figure A.47: Stacked column chart for 14 clusters and 4 hidden states in HMM

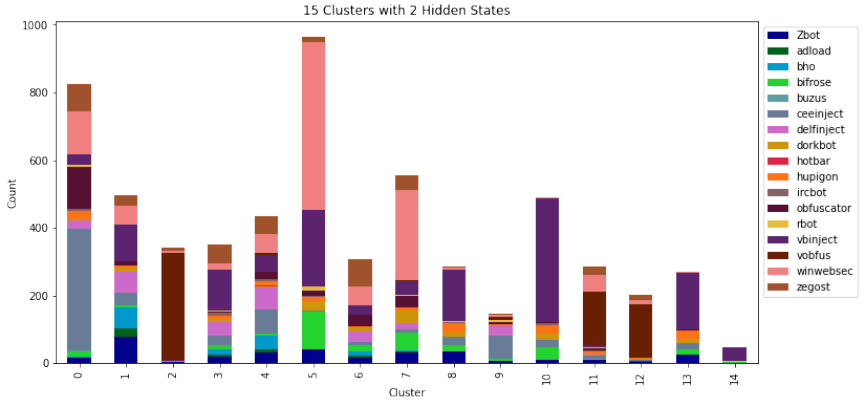

Figure A.48: Stacked column chart for 15 clusters and 2 hidden states in HMM

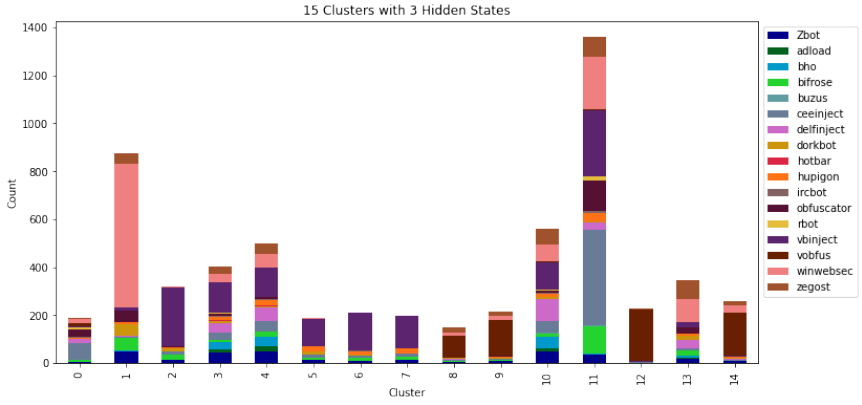

Figure A.49: Stacked column chart for 15 clusters and 3 hidden states in HMM

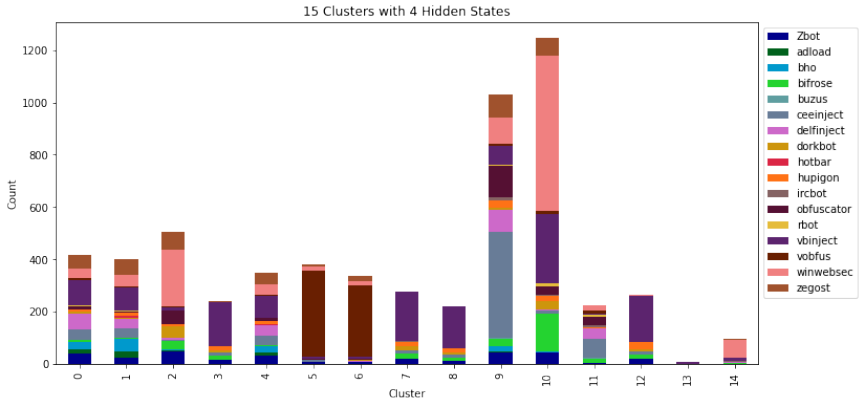

Figure A.50: Stacked column chart for 15 clusters and 4 hidden states in HMM

APPENDIX B

Graphs for K-medoids Method with Matrix Euclidean Distance

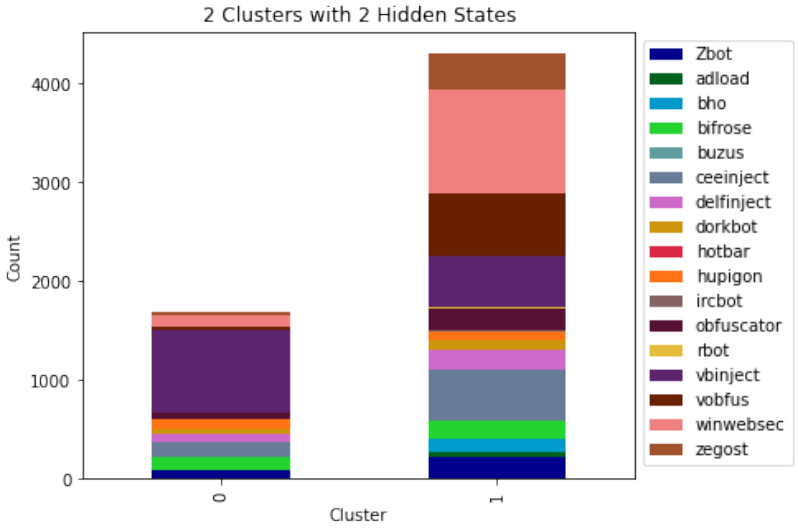

Figure B.51: Stacked column chart for 2 clusters and 2 hidden states in HMM

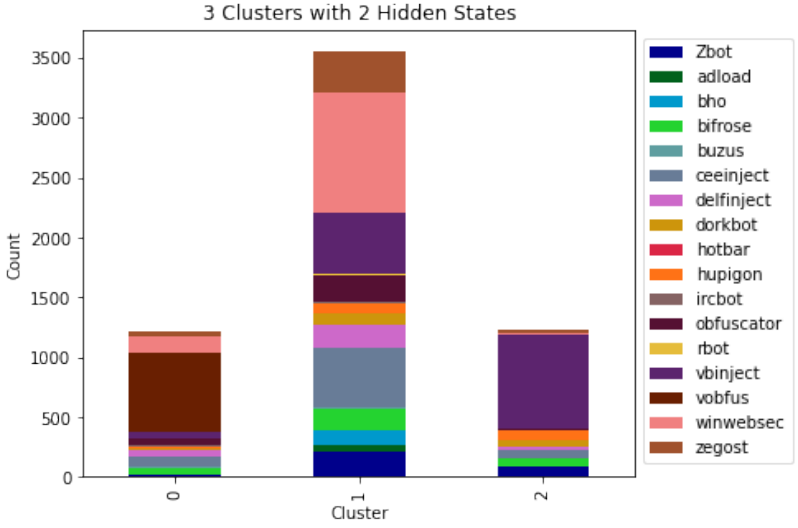

Figure B.52: Stacked column chart for 3 clusters and 2 hidden states in HMM

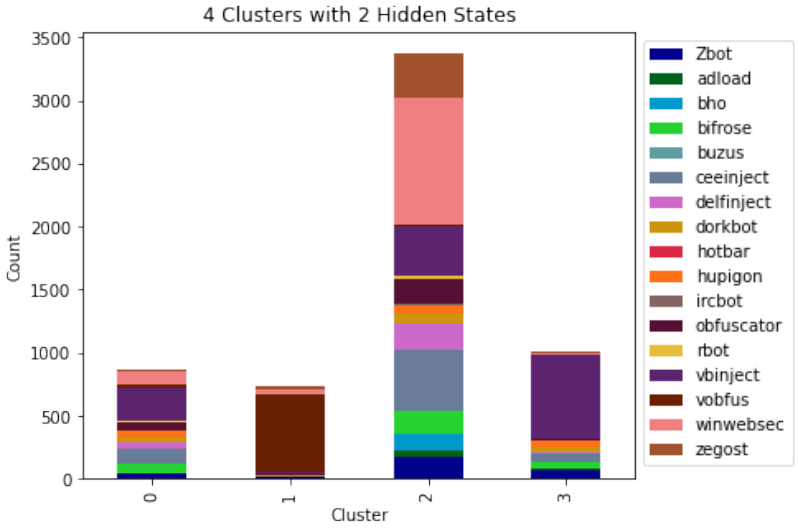

Figure B.53: Stacked column chart for 4 clusters and 2 hidden states in HMM

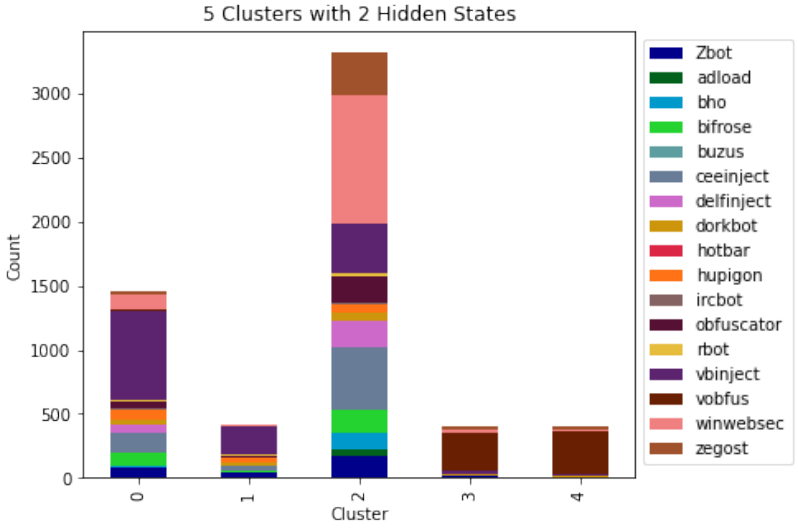

Figure B.54: Stacked column chart for 5 clusters and 2 hidden states in HMM

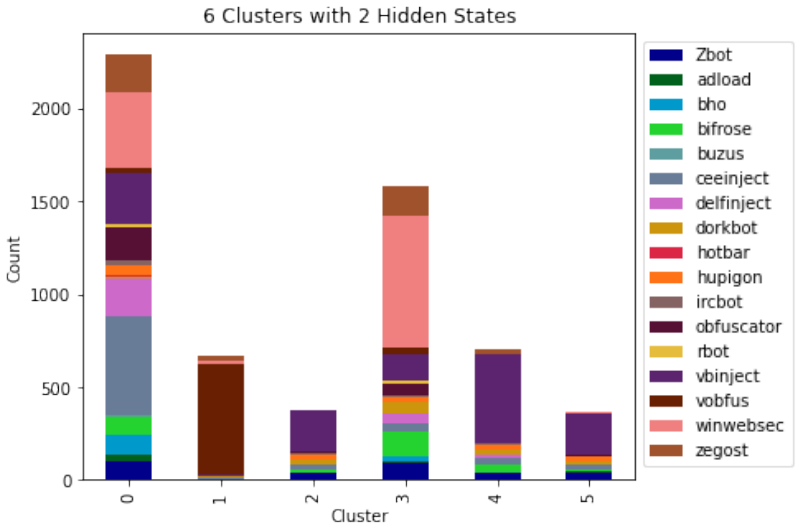

Figure B.55: Stacked column chart for 6 clusters and 2 hidden states in HMM

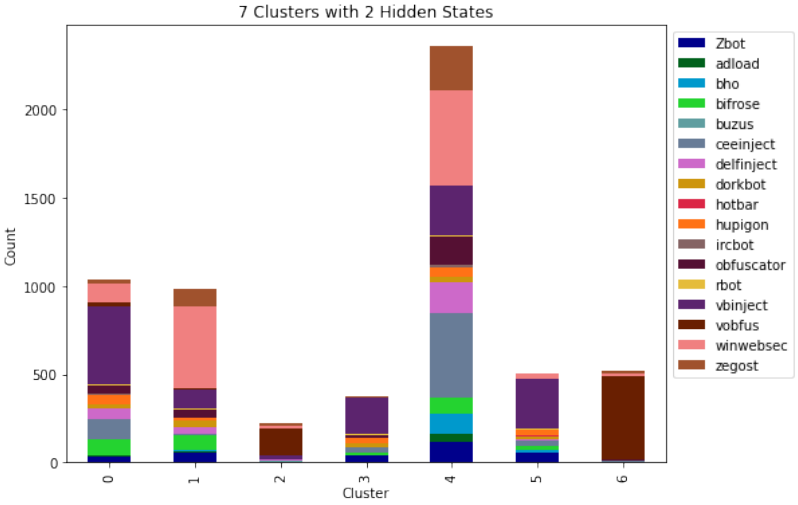

Figure B.56: Stacked column chart for 7 clusters and 2 hidden states in HMM

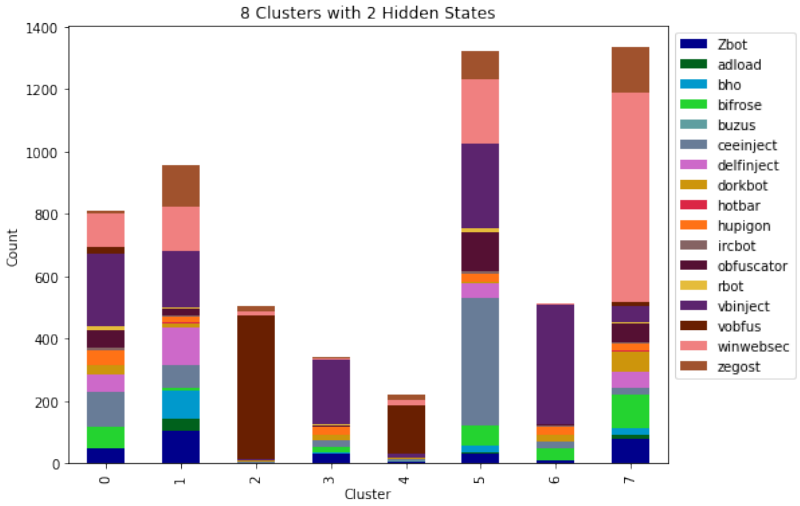

Figure B.57: Stacked column chart for 8 clusters and 2 hidden states in HMM

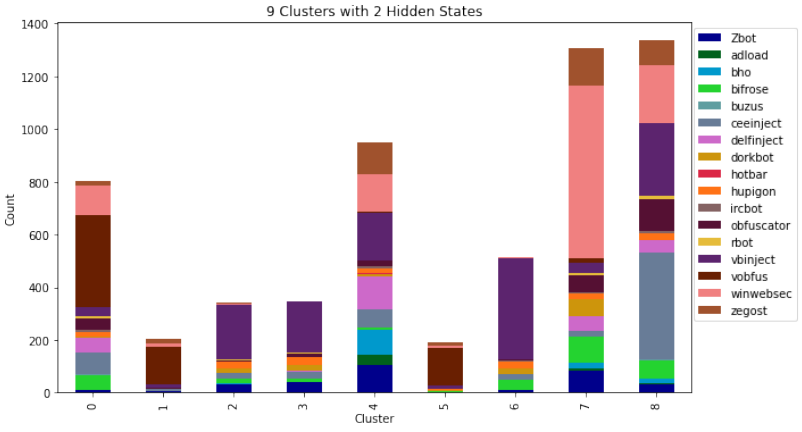

Figure B.58: Stacked column chart for 9 clusters and 2 hidden states in HMM

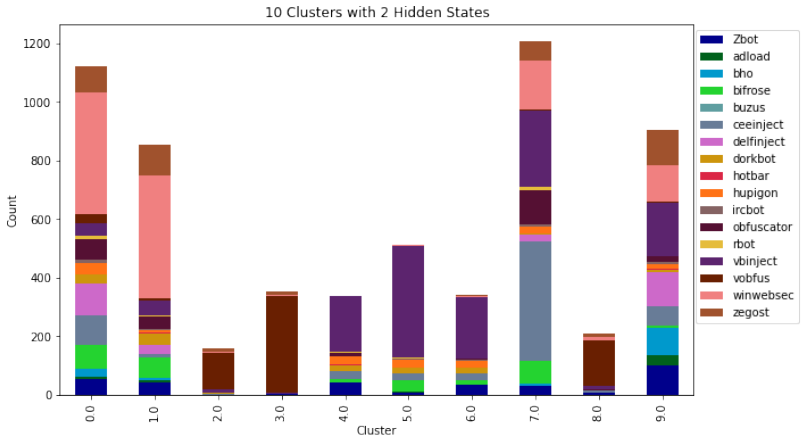

Figure B.59: Stacked column chart for 10 clusters and 2 hidden states in HMM

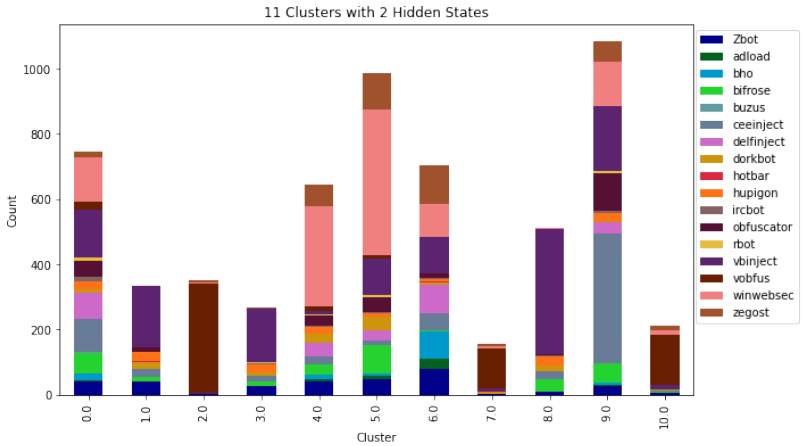

Figure B.60: Stacked column chart for 11 clusters and 2 hidden states in HMM

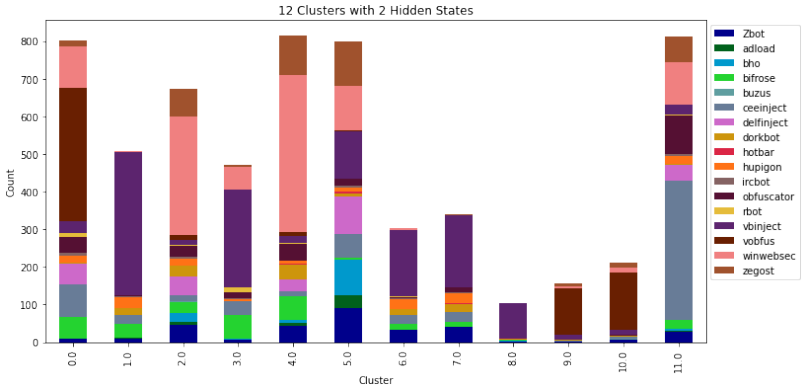

Figure B.61: Stacked column chart for 12 clusters and 2 hidden states in HMM

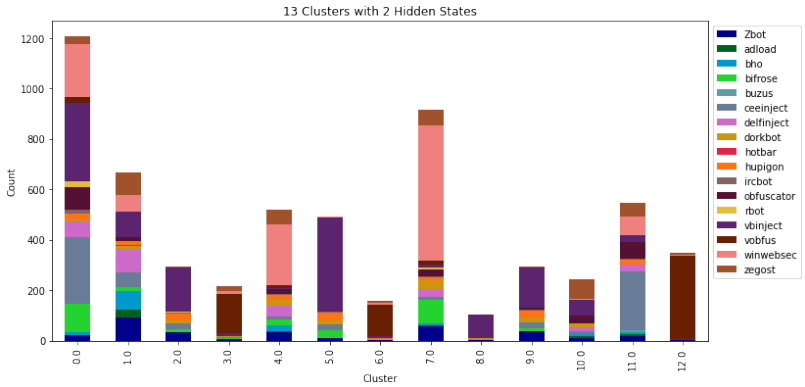

Figure B.62: Stacked column chart for 13 clusters and 2 hidden states in HMM

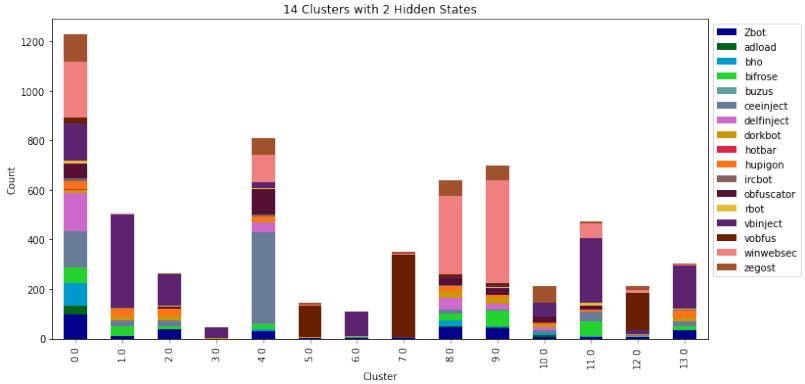

Figure B.63: Stacked column chart for 14 clusters and 2 hidden states in HMM

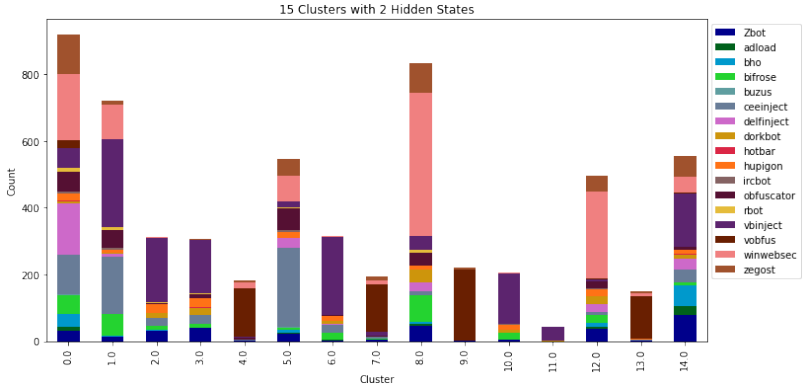

Figure B.64: Stacked column chart for 15 clusters and 2 hidden states in HMM

APPENDIX C

Naïve K-means Method for HMMs with M = 50

The naïve k-means clustering experiments were repeated for newly trained HMMs with 2 hidden states i.e. $N = 2$ and the number of observation symbols M was changed to 50 to include top 50 opcodes over the entire dataset. 92.09 percent of opcodes are represented in these HMMs. The resulting graphs and entropies of the clustering experiments are given below.

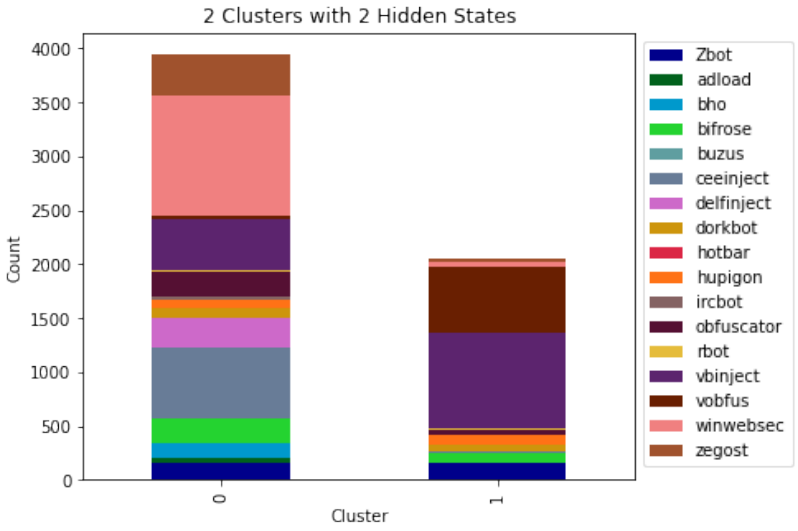

Figure C.65: Stacked column chart for 2 clusters and 2 hidden states in HMM

These experiments have been compared with kernel k-means in Appendix D.

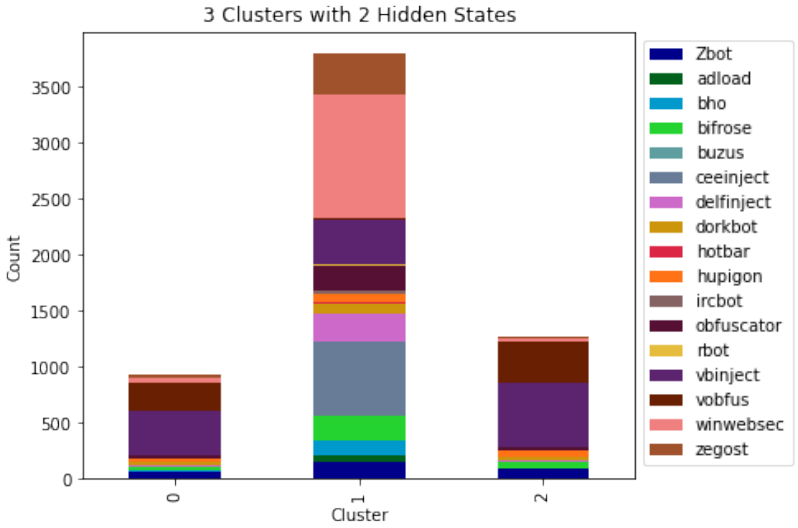

Figure C.66: Stacked column chart for 3 clusters and 2 hidden states in HMM

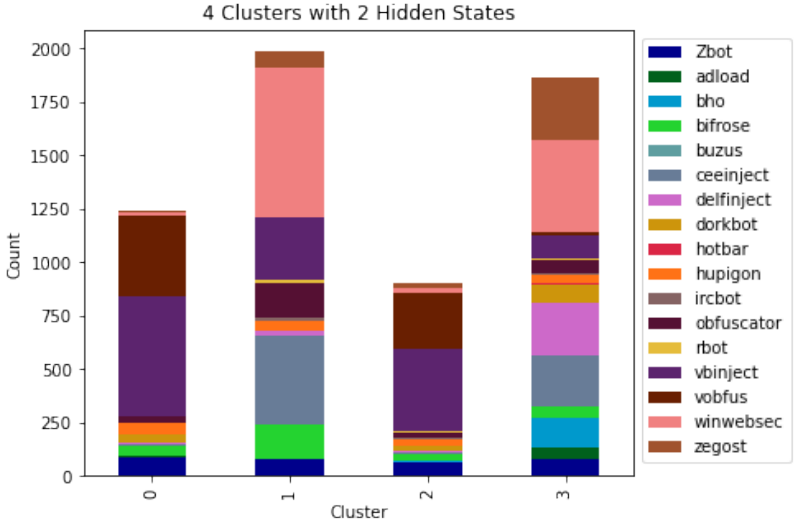

Figure C.67: Stacked column chart for 4 clusters and 2 hidden states in HMM

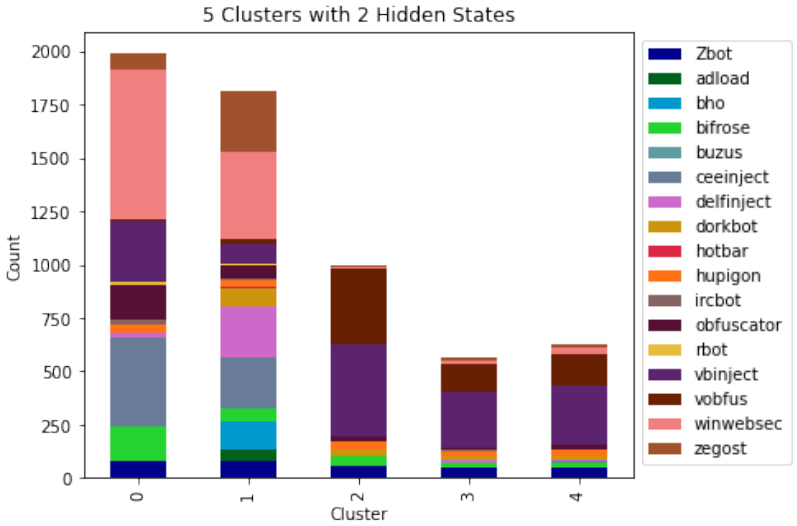

Figure C.68: Stacked column chart for 5 clusters and 2 hidden states in HMM

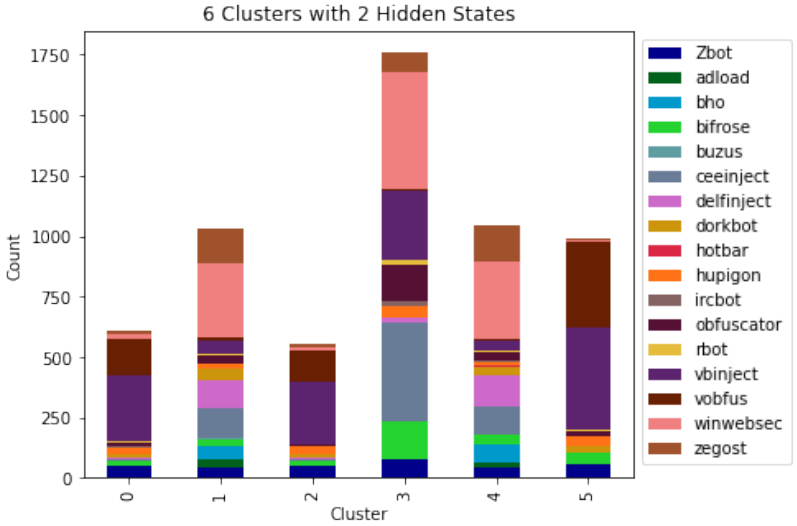

Figure C.69: Stacked column chart for 6 clusters and 2 hidden states in HMM

Figure C.70: Stacked column chart for 7 clusters and 2 hidden states in HMM

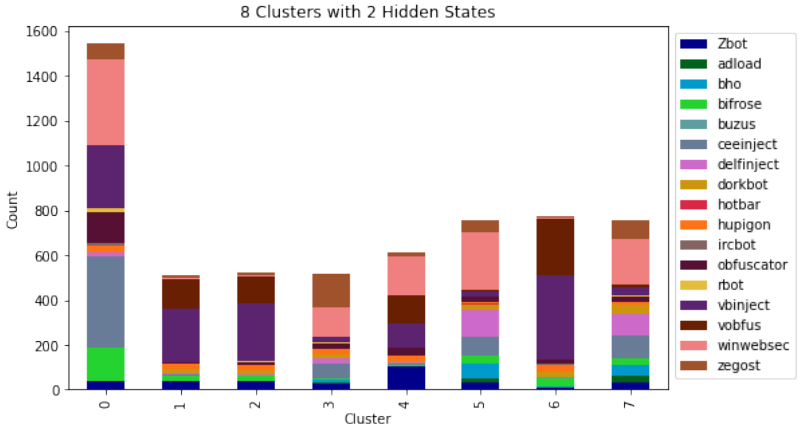

Figure C.71: Stacked column chart for 8 clusters and 2 hidden states in HMM

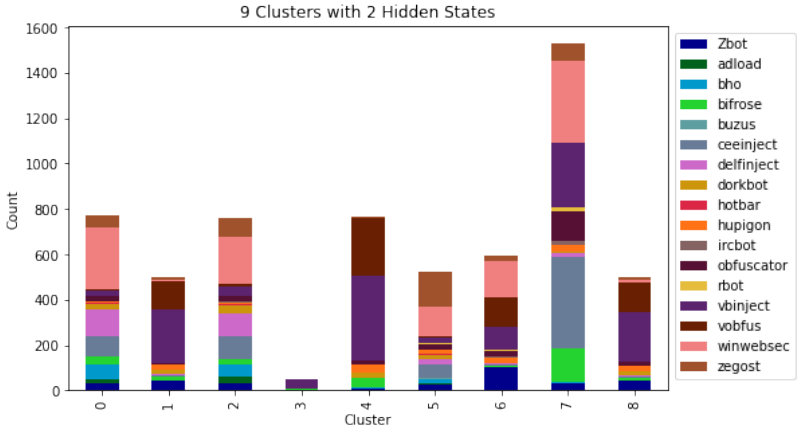

Figure C.72: Stacked column chart for 9 clusters and 2 hidden states in HMM

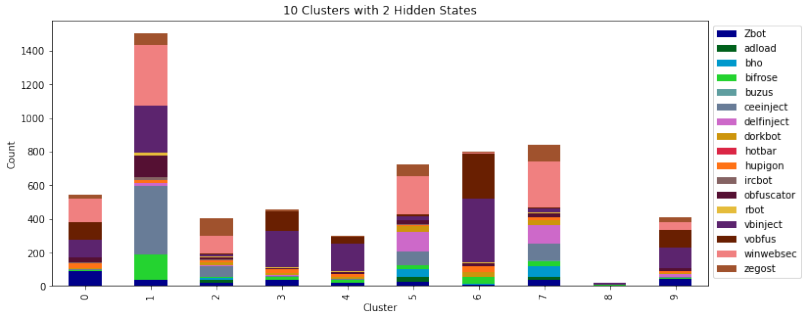

Figure C.73: Stacked column chart for 10 clusters and 2 hidden states in HMM

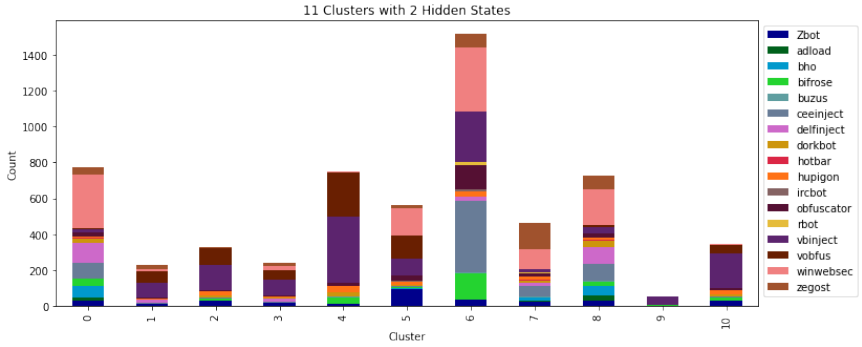

Figure C.74: Stacked column chart for 11 clusters and 2 hidden states in HMM

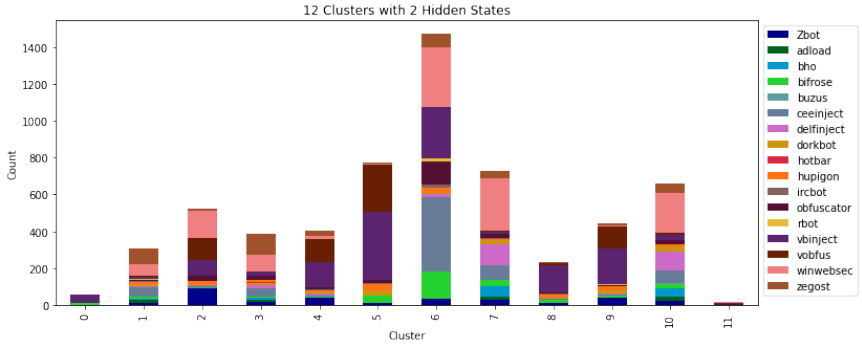

Figure C.75: Stacked column chart for 12 clusters and 2 hidden states in HMM

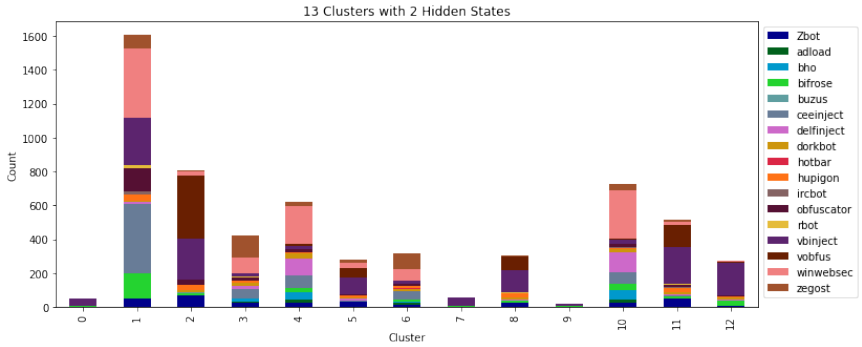

Figure C.76: Stacked column chart for 13 clusters and 2 hidden states in HMM

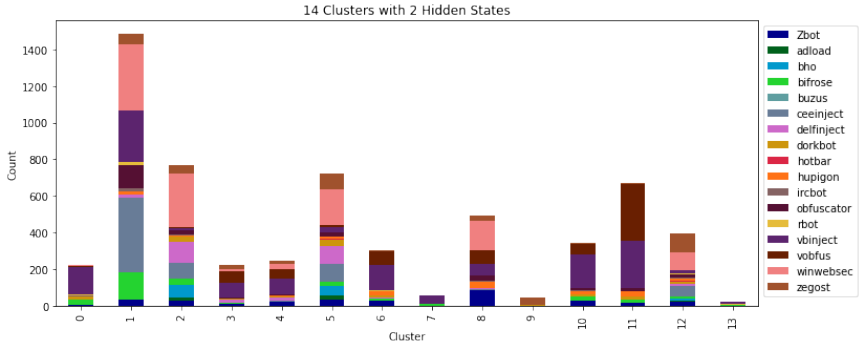

Figure C.77: Stacked column chart for 14 clusters and 2 hidden states in HMM

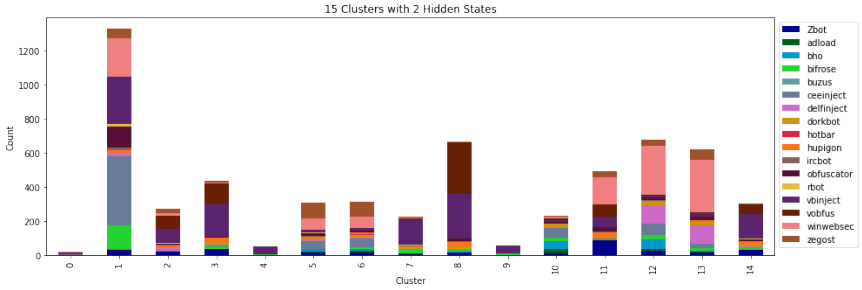

Figure C.78: Stacked column chart for 15 clusters and 2 hidden states in HMM

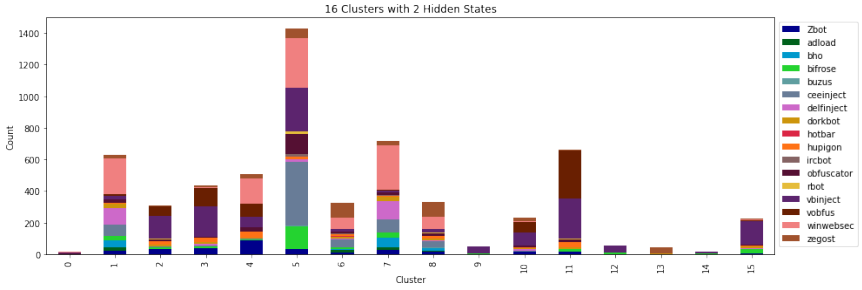

Figure C.79: Stacked column chart for 16 clusters and 2 hidden states in HMM

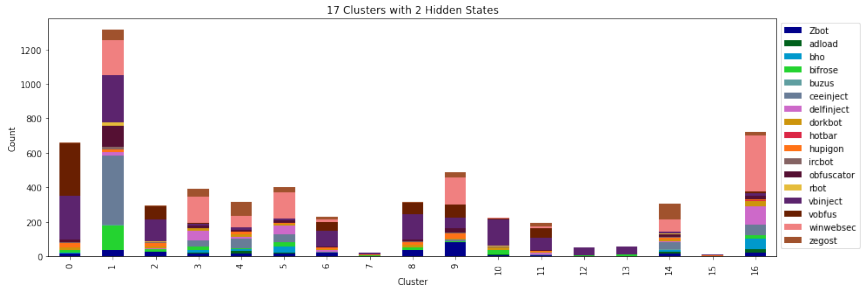

Figure C.80: Stacked column chart for 17 clusters and 2 hidden states in HMM

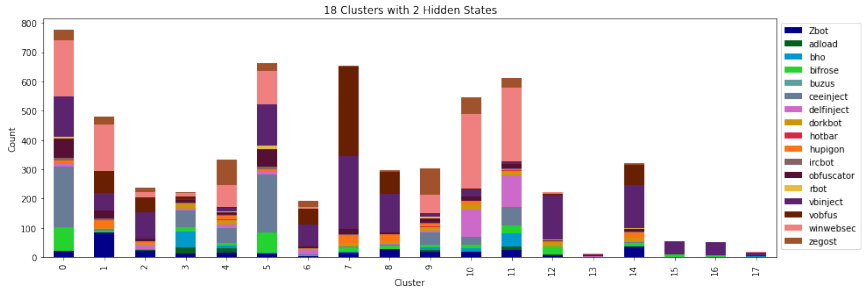

Figure C.81: Stacked column chart for 18 clusters and 2 hidden states in HMM

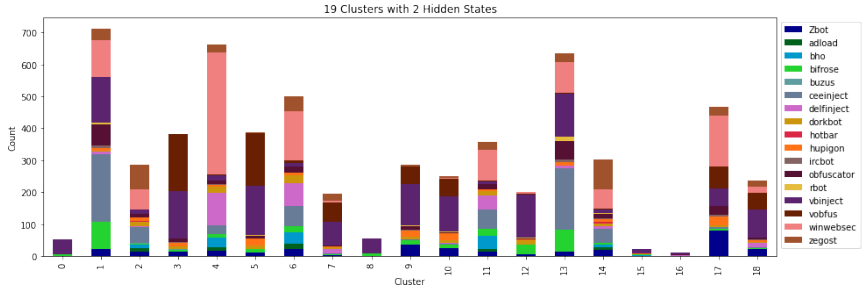

Figure C.82: Stacked column chart for 19 clusters and 2 hidden states in HMM

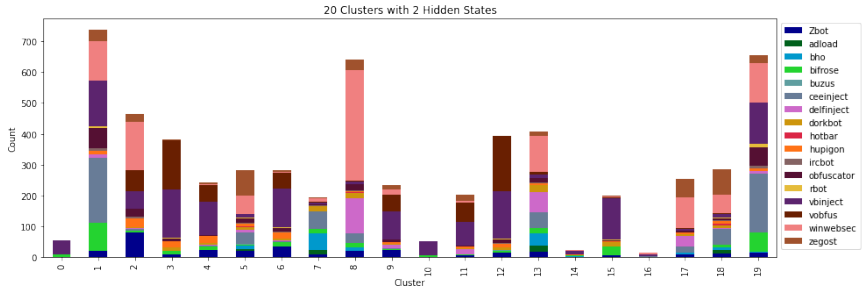

Figure C.83: Stacked column chart for 20 clusters and 2 hidden states in HMM

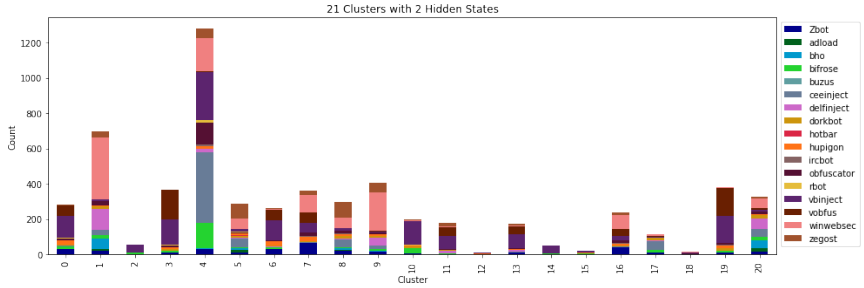

Figure C.84: Stacked column chart for 21 clusters and 2 hidden states in HMM

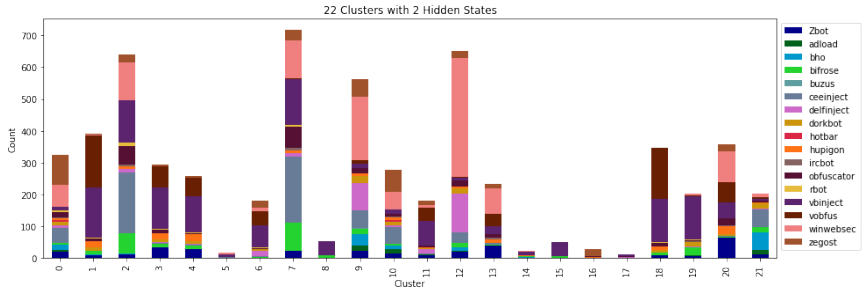

Figure C.85: Stacked column chart for 22 clusters and 2 hidden states in HMM

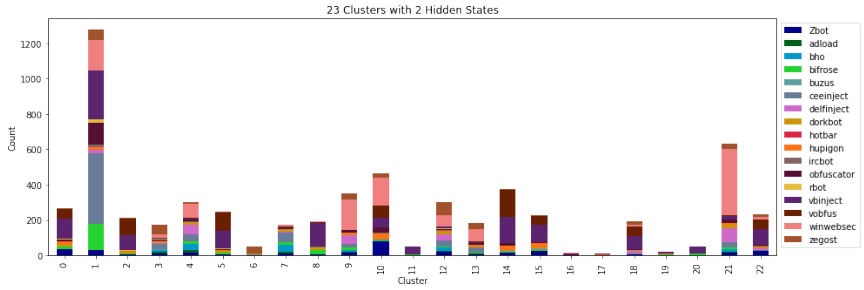

Figure C.86: Stacked column chart for 23 clusters and 2 hidden states in HMM

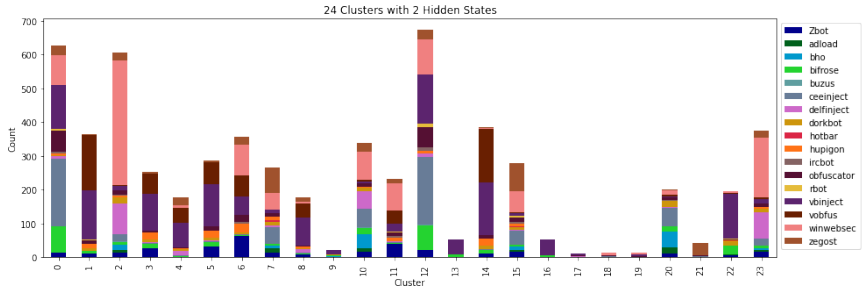

Figure C.87: Stacked column chart for 24 clusters and 2 hidden states in HMM

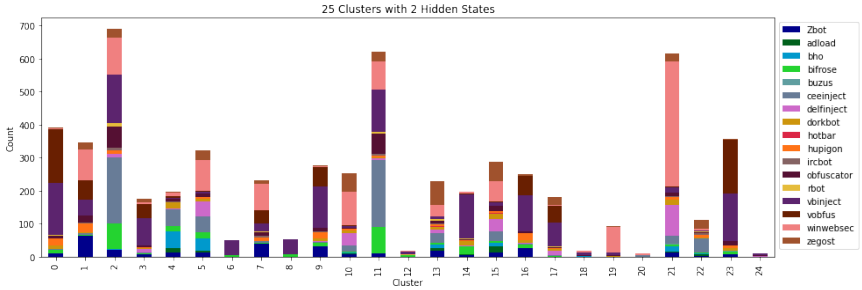

Figure C.88: Stacked column chart for 25 clusters and 2 hidden states in HMM

Table C.9: Clustering Entropy for Euclidean Distance k-means Method for all k's with $N = 2$ and $M = 50$

Cluster	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	E
$k=2$	2.247	**1.607**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2.029
$k=3$	1.717	2.226	**1.596**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2.014
$k=4$	1.553	1.895	**1.695**	2.312	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.923
$k=5$	1.897	2.309	**1.487**	1.691	1.765	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.920
$k=6$	1.723	2.244	1.699	1.971	2.209	**1.488**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.929
$k=7$	1.909	1.919	2.254	**1.380**	1.631	2.216	1.620	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.896
$k=8$	1.927	1.624	1.619	2.122	1.902	2.113	**1.375**	2.304	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.889
$k=9$	2.085	1.619	2.297	**0.357**	1.387	2.127	1.876	1.937	1.707	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.883
$k=10$	1.941	1.909	2.097	1.582	1.575	2.174	**1.394**	2.176	1.478	1.951	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.885
$k=11$	2.039	1.841	1.641	1.921	1.377	1.888	1.934	2.108	2.304	**0.510**	1.593	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.879
$k=12$	0.485	2.19	1.844	2.159	1.83	1.391	1.944	1.999	1.455	1.668	2.189	1.445	-	-	-	-	-	-	-	-	-	-	-	-	-	1.861
$k=13$	**0.363**	1.957	1.506	2.137	2.108	1.955	2.126	0.502	1.660	1.478	2.010	1.767	1.208	-	-	-	-	-	-	-	-	-	-	-	-	1.846
$k=14$	1.253	1.898	2.035	1.800	1.943	2.264	1.654	**0.502**	1.937	0.535	1.593	1.285	2.189	1.478	-	-	-	-	-	-	-	-	-	-	-	1.834
$k=15$	1.409	1.925	1.922	1.662	**0.363**	2.134	2.186	1.251	1.293	0.485	2.230	1.927	1.903	1.713	1.715	-	-	-	-	-	-	-	-	-	-	1.782
$k=16$	1.445	2.108	1.708	1.674	1.938	1.912	2.137	1.978	2.152	**0.342**	1.838	1.314	0.485	0.483	1.445	1.251	-	-	-	-	-	-	-	-	-	1.807
$k=17$	1.324	1.939	1.672	2.001	2.207	2.039	1.846	1.478	1.675	1.937	1.244	1.752	**0.363**	0.477	2.118	1.541	1.908	-	-	-	-	-	-	-	-	1.810
$k=18$	1.900	1.913	1.897	2.147	2.191	1.936	1.710	1.316	1.669	2.175	1.811	1.925	1.247	1.541	1.669	0.502	**0.302**	1.409	-	-	-	-	-	-	-	1.793
$k=19$	**0.392**	1.924	2.170	1.300	1.597	1.366	2.198	1.682	0.498	1.702	1.693	2.198	1.241	1.948	2.154	1.478	1.541	1.921	1.885	-	-	-	-	-	-	1.796
$k=20$	0.502	1.930	1.926	1.360	1.672	2.187	1.755	2.012	1.578	1.838	**0.378**	1.713	1.309	2.200	1.478	1.237	1.363	1.783	2.149	1.927	-	-	-	-	-	1.770
$k=21$	1.725	1.724	0.482	1.295	1.909	2.126	1.681	1.956	2.175	1.655	1.220	1.634	1.541	1.671	**0.363**	1.529	1.941	1.829	1.330	1.386	2.320	-	-	-	-	1.762
$k=22$	2.185	1.372	1.912	1.698	1.703	1.330	1.702	1.919	0.487	2.103	2.224	1.746	1.507	1.915	1.529	**0.363**	0.640	1.541	1.275	1.265	1.950	2.043	-	-	-	1.763
$k=23$	1.736	1.927	1.285	2.230	2.211	1.355	0.643	2.017	1.249	1.745	1.919	**0.383**	2.258	1.902	1.372	1.716	1.330	1.541	1.703	1.529	0.487	1.603	1.879	-	-	1.755
$k=24$	1.871	1.273	1.502	1.701	1.686	1.710	1.961	2.188	1.682	1.529	2.189	1.904	1.952	0.482	1.400	2.124	**0.303**	1.541	1.073	0.992	2.080	0.483	1.220	1.765	-	1.739
$k=25$	1.404	1.960	1.963	1.676	2.072	2.124	**0.360**	1.913	0.482	1.714	1.814	1.855	1.306	2.250	1.233	2.287	1.702	1.664	1.445	0.719	0.935	1.465	2.043	1.253	1.541	1.737

APPENDIX D

Kernel K-means Method for HMMs with M = 50

The kernel k-means clustering experiments were repeated for newly trained HMMs with 2 hidden states i.e. $N = 2$ and the number of observation symbols M was changed to 50 to include top 50 opcodes over the entire dataset. Kernel k-means method involved bootstrapping the matrix Euclidean distance function with the original k-means clustering algorithm. The resulting graphs and entropies of the clustering experiments are given below.

We also compared the results of all our experiments with 50-opcode HMMs, from both Appendix C and Appendix D by comparing the weighted intra-cluster entropies for all values of k from 2 to 25, and displayed the result as a graph given the Figure D.113. It shows decreasing trends for both the experiments, with increase in the number of clusters k.

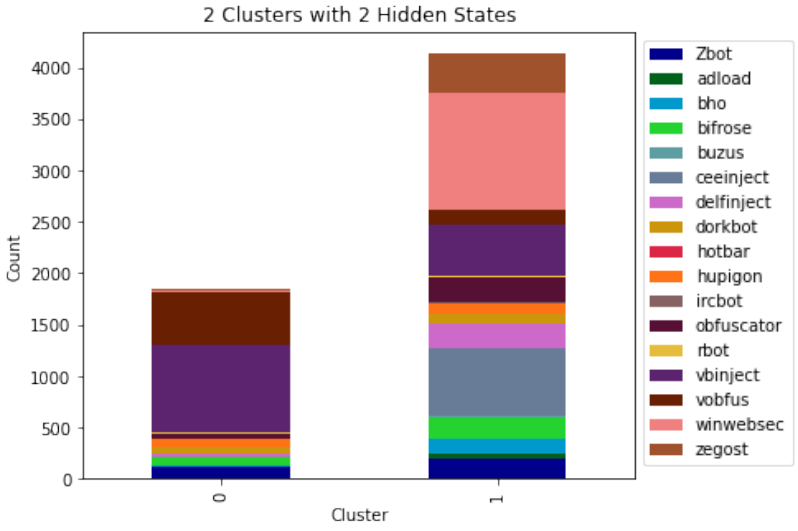

Figure D.89: Stacked column chart for 2 clusters and 2 hidden states in HMM

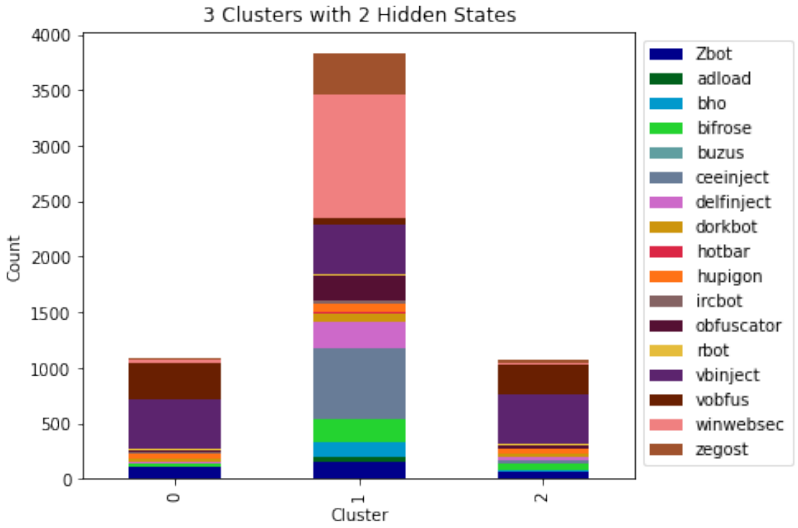

Figure D.90: Stacked column chart for 3 clusters and 2 hidden states in HMM

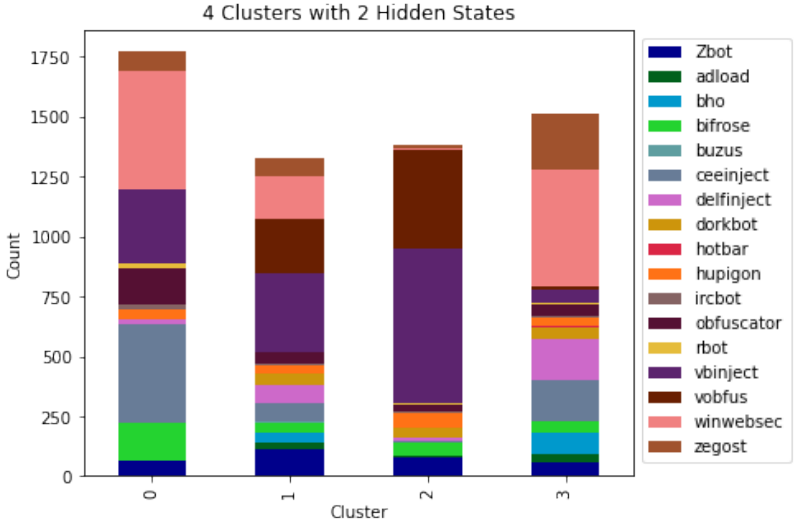

Figure D.91: Stacked column chart for 4 clusters and 2 hidden states in HMM

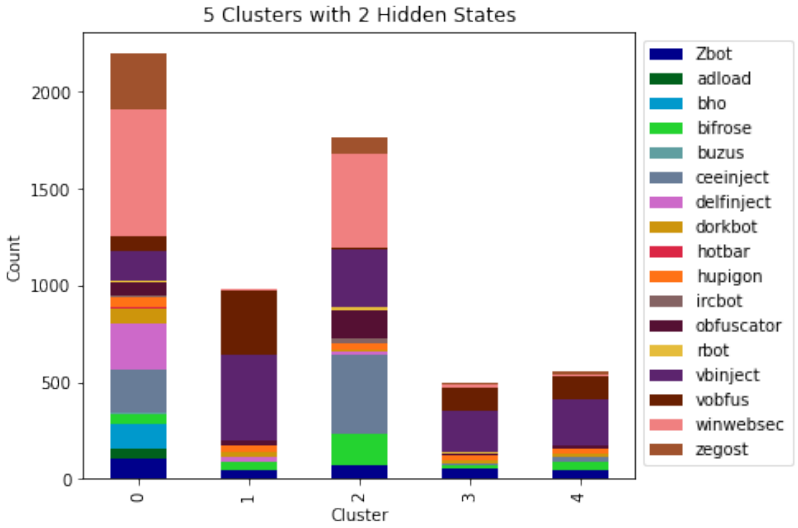

Figure D.92: Stacked column chart for 5 clusters and 2 hidden states in HMM

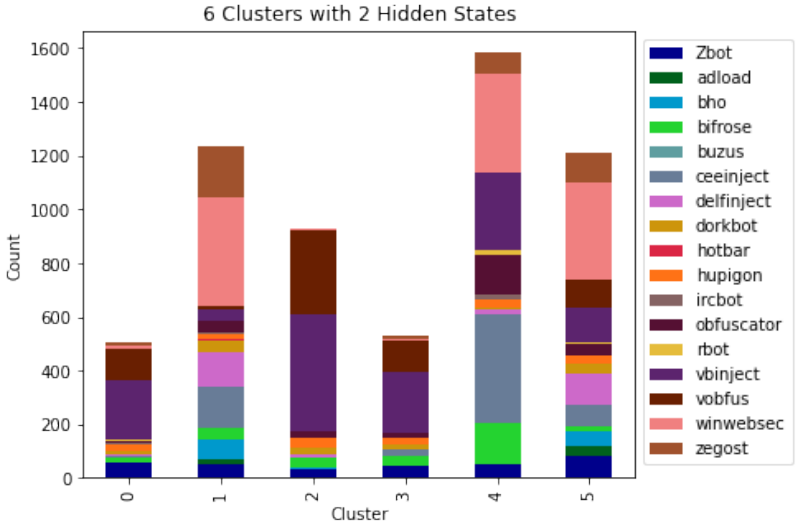

Figure D.93: Stacked column chart for 6 clusters and 2 hidden states in HMM

Figure D.94: Stacked column chart for 7 clusters and 2 hidden states in HMM

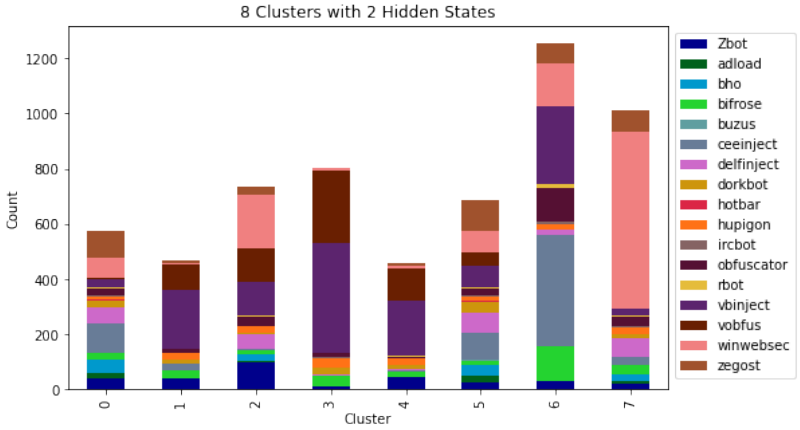

Figure D.95: Stacked column chart for 8 clusters and 2 hidden states in HMM

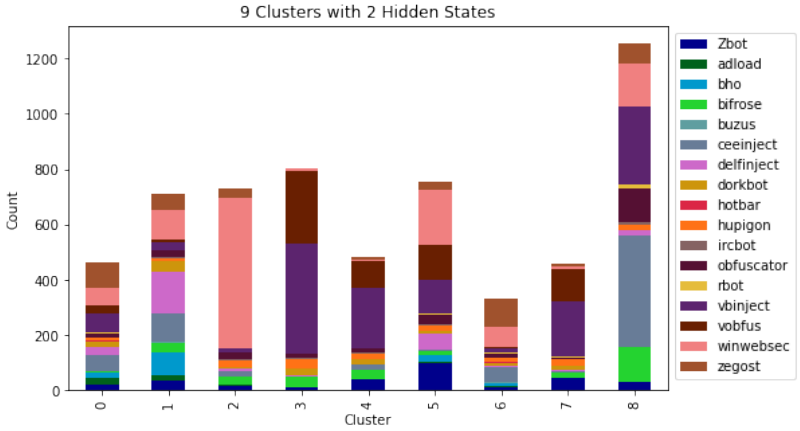

Figure D.96: Stacked column chart for 9 clusters and 2 hidden states in HMM

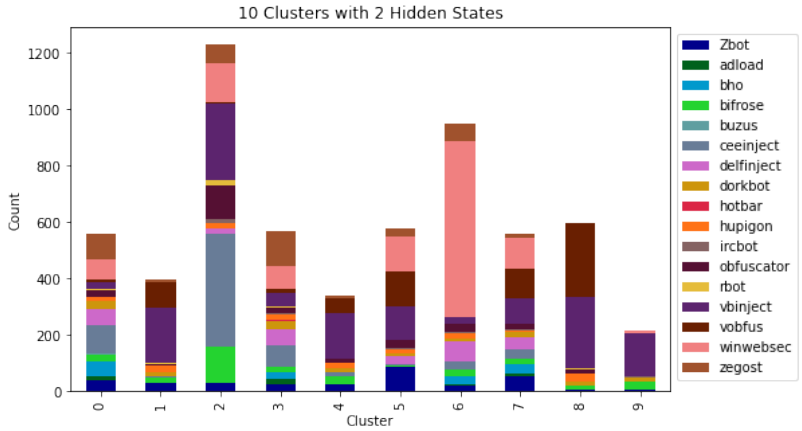

Figure D.97: Stacked column chart for 10 clusters and 2 hidden states in HMM

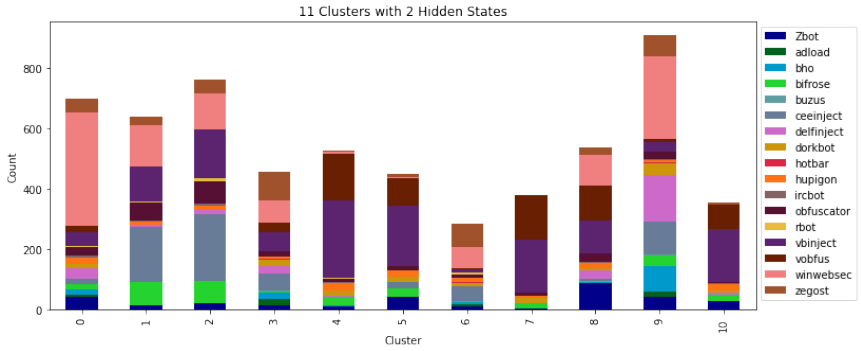

Figure D.98: Stacked column chart for 11 clusters and 2 hidden states in HMM

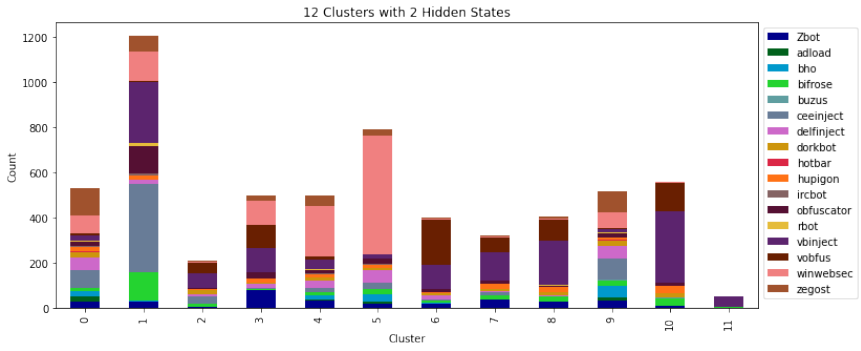

Figure D.99: Stacked column chart for 12 clusters and 2 hidden states in HMM

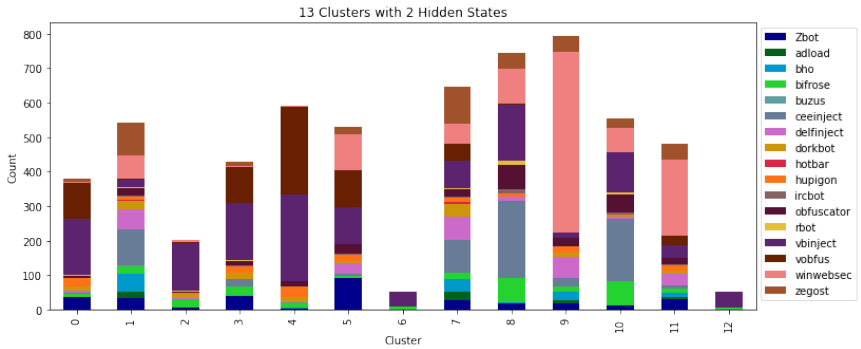

Figure D.100: Stacked column chart for 13 clusters and 2 hidden states in HMM

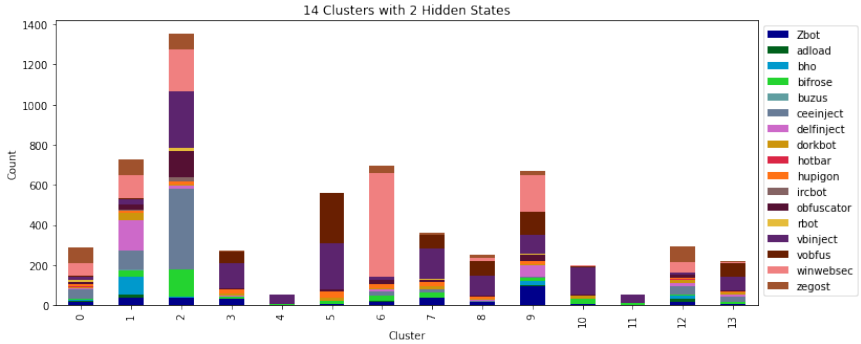

Figure D.101: Stacked column chart for 14 clusters and 2 hidden states in HMM

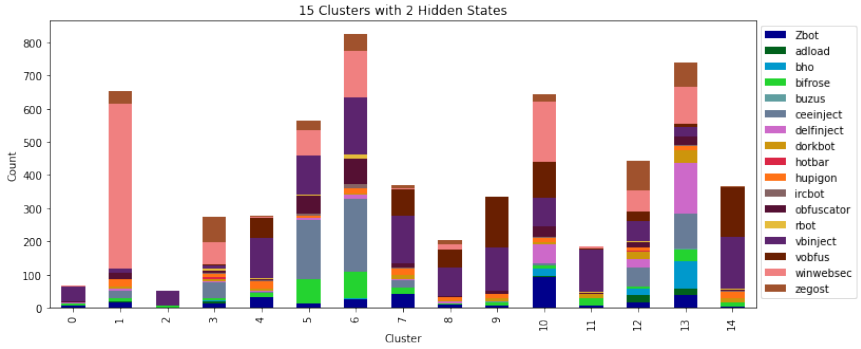

Figure D.102: Stacked column chart for 15 clusters and 2 hidden states in HMM

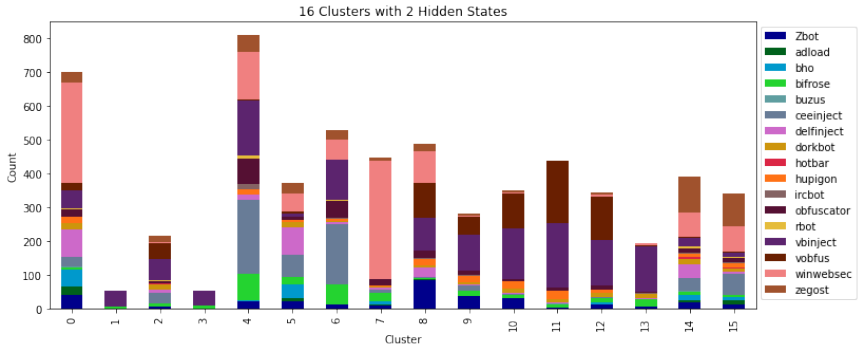

Figure D.103: Stacked column chart for 16 clusters and 2 hidden states in HMM

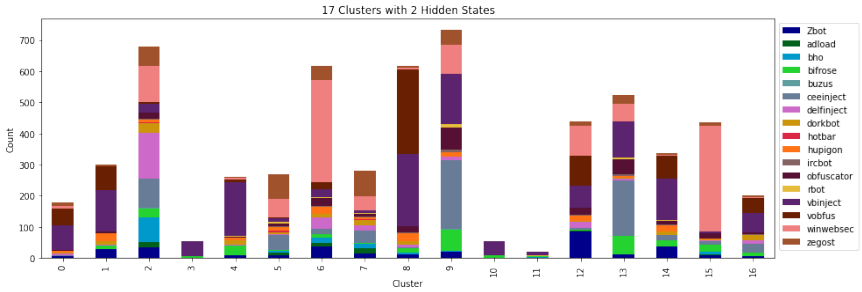

Figure D.104: Stacked column chart for 17 clusters and 2 hidden states in HMM

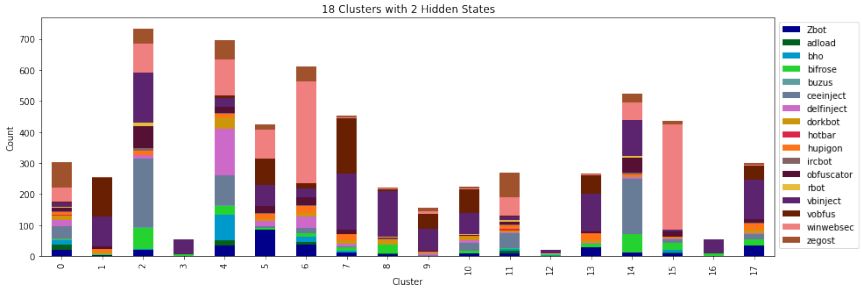

Figure D.105: Stacked column chart for 18 clusters and 2 hidden states in HMM

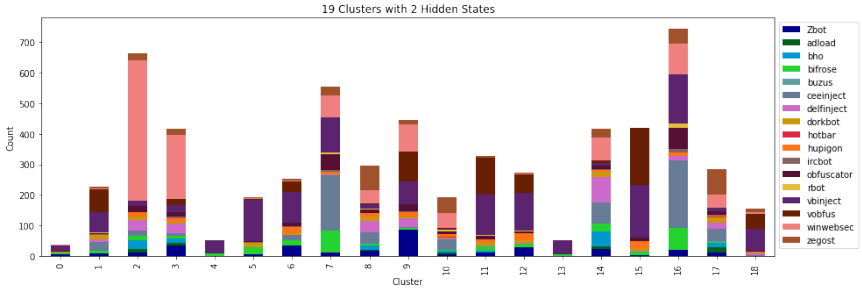

Figure D.106: Stacked column chart for 19 clusters and 2 hidden states in HMM

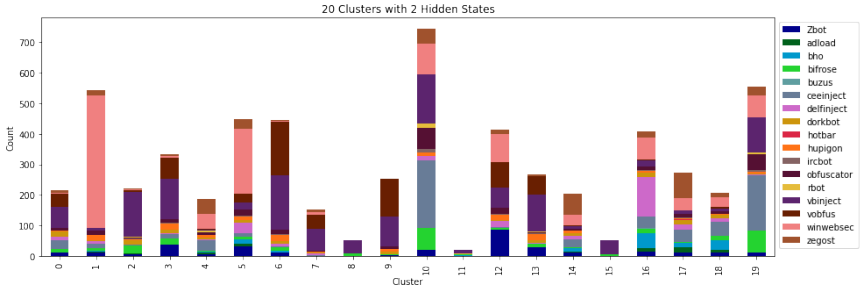

Figure D.107: Stacked column chart for 20 clusters and 2 hidden states in HMM

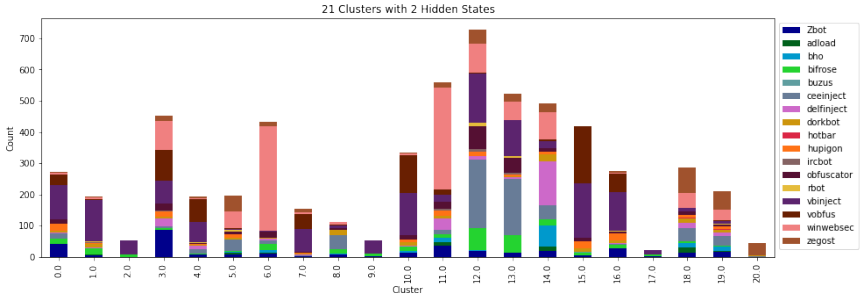

Figure D.108: Stacked column chart for 21 clusters and 2 hidden states in HMM

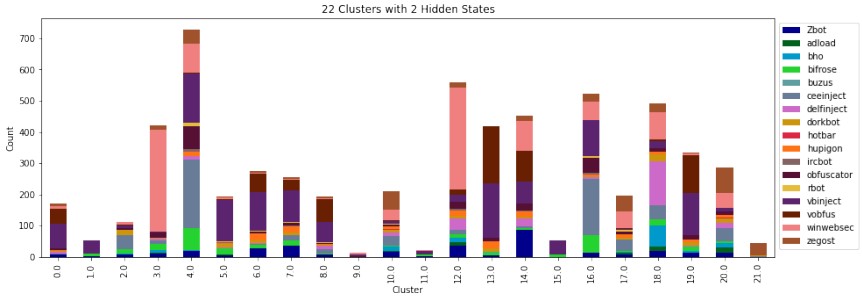

Figure D.109: Stacked column chart for 22 clusters and 2 hidden states in HMM

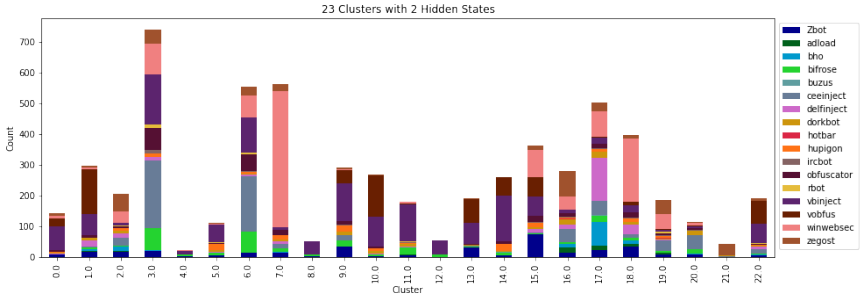

Figure D.110: Stacked column chart for 23 clusters and 2 hidden states in HMM

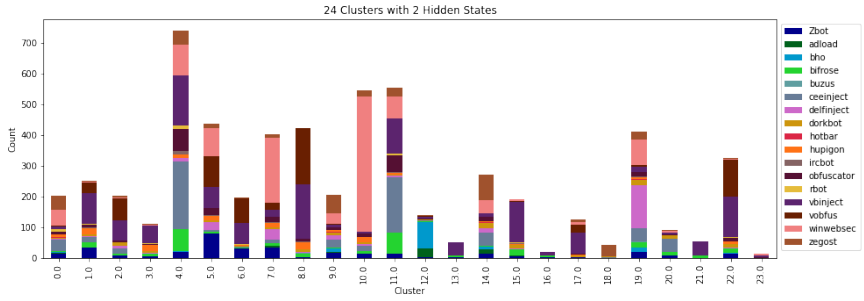

Figure D.111: Stacked column chart for 24 clusters and 2 hidden states in HMM

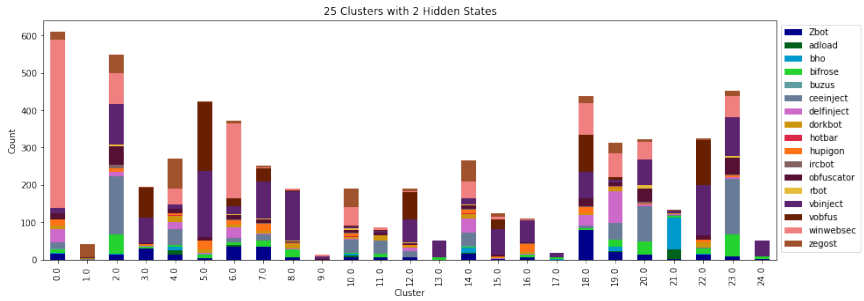

Figure D.112: Stacked column chart for 25 clusters and 2 hidden states in HMM

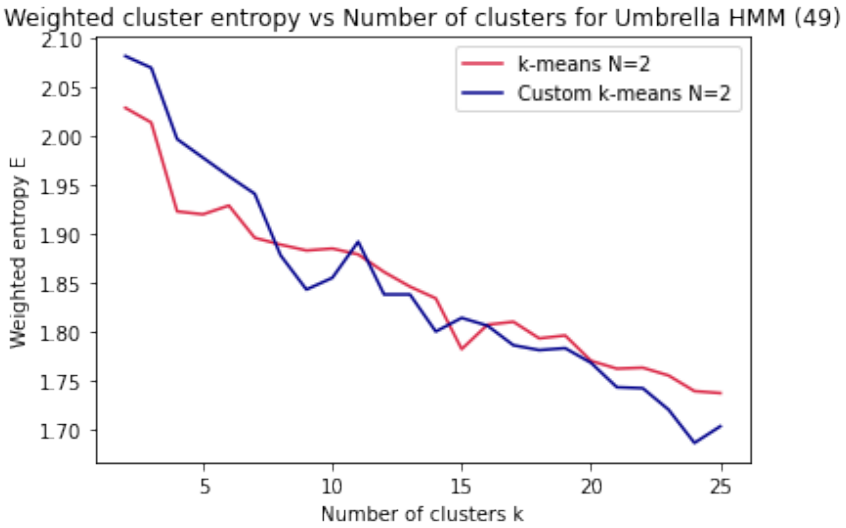

Figure D.113: Comparison of Weighted Entropies as a function of k

Table D.10: Clustering Entropy for Euclidean Distance k-means Method for all k's with $N = 2$ and $M = 50$

Cluster	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	E
$k=2$	**1.628**	2.285	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2.082
$k=3$	**1.684**	2.250	1.816	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2.070
$k=4$	1.957	2.310	**1.544**	2.183	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.997
$k=5$	2.286	**1.517**	1.980	1.732	1.792	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.978
$k=6$	1.743	2.156	**1.456**	1.773	1.976	2.292	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.959
$k=7$	1.772	2.420	1.687	1.954	**1.359**	2.105	2.098	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.941
$k=8$	2.405	1.739	2.118	**1.359**	1.690	2.453	1.929	1.513	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.878
$k=9$	2.401	2.320	1.158	**1.359**	1.749	2.127	2.108	1.690	1.929	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.843
$k=10$	2.404	1.594	1.930	2.419	1.742	2.001	1.432	2.284	**1.255**	1.173	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.855
$k=11$	1.811	1.872	1.982	2.342	1.519	1.764	2.102	**1.281**	2.033	2.174	1.580	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1.892
$k=12$	2.386	1.921	1.990	1.985	2.024	1.404	1.562	1.855	1.583	2.371	1.400	**0.449**	-	-	-	-	-	-	-	-	-	-	-	-	-	1.838
$k=13$	1.692	2.378	1.290	1.833	1.276	2.028	0.515	2.465	1.976	1.414	1.864	1.973	**0.449**	-	-	-	-	-	-	-	-	-	-	-	-	1.838
$k=14$	2.164	2.319	1.966	1.680	**0.451**	1.243	1.143	1.817	1.751	2.111	1.265	0.528	2.236	1.846	-	-	-	-	-	-	-	-	-	-	-	1.800
$k=15$	1.085	1.065	**0.449**	2.115	1.709	1.885	2.018	1.866	1.708	1.316	2.096	1.188	2.387	2.327	1.304	-	-	-	-	-	-	-	-	-	-	1.814
$k=16$	2.042	0.549	2.027	**0.515**	2.004	2.265	1.854	1.004	2.017	1.891	1.647	1.260	1.569	1.226	2.287	2.118	-	-	-	-	-	-	-	-	-	1.806
$k=17$	1.600	1.662	2.283	**0.445**	1.302	2.098	1.832	2.237	1.443	1.976	0.515	1.364	1.990	1.855	1.844	1.003	1.972	-	-	-	-	-	-	-	-	1.786
$k=18$	2.292	1.191	1.976	**0.459**	2.301	2.005	1.817	1.529	1.305	1.459	1.814	2.098	1.364	1.675	1.855	1.003	0.515	1.867	-	-	-	-	-	-	-	1.781
$k=19$	1.729	1.862	1.345	1.907	0.528	1.205	1.913	1.863	2.245	2.027	2.100	1.540	1.681	**0.429**	2.261	1.244	1.978	2.234	1.436	-	-	-	-	-	-	1.783
$k=20$	2.055	0.961	1.305	1.840	2.096	1.961	1.515	1.454	0.515	1.191	1.978	1.364	2.007	1.675	2.145	**0.455**	2.172	2.227	2.285	1.863	-	-	-	-	-	1.768
$k=21$	1.925	1.205	**0.442**	2.027	1.682	2.101	1.016	1.452	1.921	0.528	1.560	1.715	1.973	1.855	2.195	1.244	1.681	1.364	2.253	2.250	0.483	-	-	-	-	1.743
$k=22$	1.599	0.528	1.921	0.979	1.973	1.205	1.681	1.906	1.682	1.012	2.250	1.400	1.715	1.244	2.027	0.449	1.855	2.101	2.195	1.560	2.253	0.483	-	-	-	1.742
$k=23$	1.555	1.613	2.238	1.983	1.364	1.623	1.863	0.990	0.500	1.873	1.199	1.255	0.494	1.403	1.303	1.992	2.236	2.215	1.850	2.111	1.921	**0.483**	1.676	-	-	1.720
$k=24$	2.188	1.913	1.765	1.623	1.983	2.025	1.408	1.783	1.257	2.225	0.909	1.863	1.297	0.500	2.205	1.207	1.400	1.387	0.493	2.072	1.732	**0.449**	1.523	1.012	-	1.686
$k=25$	1.168	0.493	2.011	1.408	2.212	1.257	1.764	1.913	1.207	1.012	2.094	1.831	1.680	**0.469**	2.312	1.387	1.623	1.267	2.036	2.130	1.917	1.240	1.523	1.825	0.500	1.703