A HIGHER ORDER COLLECTIVE CLASSIFIER

by

VIKAS MENON

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Computer Science

Written under the direction of

Paul B. Kantor and

Co-Directed by William M. Pottenger

and approved by

_____

_____

_____

New Brunswick, New Jersey

May, 2009

ABSTRACT OF THE THESIS

A Higher Order Collective Classifier

By Vikas Menon

Thesis Director:
Paul B. Kantor and

Co-Directed by William M. Pottenger

Modern statistical machine learning techniques often rely on the assumption that data instances are independent and identically distributed (IID). However, recent work in statistical relational learning has demonstrated the utility of violating the independence assumption. Specifically, the research has shown the value of leveraging relationships between data instances based on higher-order paths. In this thesis, I present a novel Higher Order Collective Classifier (HOCC), a statistical relational machine learning technique that leverages latent information present in higher-order co-occurrences of items across data instances. A general framework is presented in which HOCC can be applied to event detection in time series data. Given the importance of cyber-security, HOCC is applied to two different data sets in the cyber-security domain: first, a Border Gateway Protocol (BGP) dataset, for detection and classification of anomalies, and second, a Network File System dataset for building models of user activity for masquerade detection. Performance of HOCC compares favorably against first-order models that do not leverage higher-order information, achieving separation of classes that heretofore were difficult to separate.

Acknowledgements

My experience in the Department of Computer Science at Rutgers University has been nothing short of life changing and I have a number of people to thank for it.

First and foremost, I wish to thank Prof. William Pottenger, without whom this work would not have been possible. I want to thank him in particular for his words of encouragement, support and his belief in me. He has been through my ups and more importantly downs and gave me some much needed advice during times of need.

I would like to thank the members on my defense committee, Prof. Paul Kantor, Prof. Liviu Iftode and Prof. Amélie Marian for taking time out of their busy schedules to oversee my defense.

Rutgers has been very kind to me and I owe it to a lot of people here including Prof. Endre Szemeredi, Prof. Uli Kremer, Prof. Ricardo Bianchini, Prof. Ken Shan, Prof. Amélie Marian (again :) and Prof. Michael Grigoriadis for teaching me Computer Science. I wish to thank Carol DiFrancesco for helping out with the tons of paper work I churned out for her. I also thank Nikita and Stephen for very insightful discussions in my field.

When, I came to Rutgers, I was helped by a number of graduate students who were part of the Rutgers Indian Graduate Student Association (RIGSA) and thank them for being around and lending a helping hand when I needed it the most.

Last, but definitely not the least, I want to thank my extended family, my roommates and friends, my home away from home; Parashar, Teda, Rohit and Mona.

Dedication

I dedicate this thesis to Mom, Dad, Chechi and Bala.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The focus of the applications in this thesis is, network (or cyber) security. Today a major fraction of financial transactions happen on the Internet, and it is estimated that companies worldwide lost over a trillion dollars in 2008 due to security breaches and/or masquerades. Recently, for example, a $250,000 bounty was offered by Microsoft to obtain information leading to the arrest of the creator of a difficult to eradicate worm, Conficker. In general, addressing a problem in cyber-security involves three steps: detection, classification and resolution. The first two steps are important for sound diagnosis ultimately leading to resolution.

The primary motivation for this work has been the difficulty in distinguishing between different classes of events in network data. In this thesis, we present a novel Higher Order Collective Classifier (HOCC) capable of distinguishing between varieties of network events, in particular applied to security.Though the focus is on network security applications, I believe that HOCC can be used to solve a large variety of problems related to classification of time series data.

We solve two distinct and difficult network security problems; first anomaly detection and classification and second, masquerade detection. The goal of anomaly detection and classification systems is to first identify a disruptive or unusual event (anomaly detection) and then provide some analysis or background by comparing the ongoing problem event

with past problem events (classification). Identifying non-normal network activity is called anomaly detection, while classification is the task of assigning a class or label to an instance based on domain expertise. Both anomaly detection and classification are necessary for providing a sound diagnosis and resolution of a disruptive event. Another way of looking at the same problem is by identifying ongoing activity on the network. Such activity may be normal, worm, electricity loss and so on. These events in network data can be separated using two distinguishing criteria. First, binary, where each event can be classified as either anomalous or normal or in other words, anomaly detection. Second, general multinomial classification, where an event can be classified to belong to more than one class from several, such as network outage, virus, electricity loss, etc, or simply put, classification. No system has been capable of both anomaly detection and classification in particular to the Border Gateway Protocol (BGP). We discuss this problem in more detail in Chapter 4.

The other class of network security problems that requires separation is masquerade detection. Masquerades are attempts by users (potentially with legitimate system access) to gain privileges not available to them, such as a user obtaining the administrator password to gain access to files that would otherwise not be accessible to him/her. Masquerade detection has traditionally been a difficult problem.

One reason why masquerade detection is a difficult problem is the lack of sufficiently large labeled data. Previous efforts in masquerade detection have relied on a single dataset, the Schonlau dataset. Given the limited amount of datasets to test masquerade detection systems, it is understandable that present systems are not robust enough for large scale deployment. In our efforts to test for masquerades, we worked on a previously unexplored dataset from the perspective of masquerade detection. Our success at solving this problem indicates that a new and large dataset now has the potential to be utilized for the

advancement of masquerade detection systems.

1.1 A Brief Overview

To begin a more detailed discussion we give a very abstract view on the working of the Higher Order Collective Classifier (HOCC). HOCC essentially generates a distribution of integers from a graph generated over a window of time from the network data. The generated distribution can be compared with another window's distribution using various statistical tests. We use the Student's T-test p-values to differentiate between distributions generated from two distinct windows. The Student's T-test p-value provides a confidence measure and is a number between zero and one. A p-value of less than 0.05 is sufficient to conclude that two windows are distinctly different from one another. The ability to distinguish between events and separate them is of key importance to the problem of classification. Using this simplistic framework, we show how HOCC can be applied to a large class of problems. In this thesis our primary focus is on the class of network security problems. We discuss the details of the working of HOCC in Chapter 3.

As mentioned above, HOCC deals with graphs, the obvious next question is how are these graphs generated? The answer lies in the use of co-occurrence information. Co-occurrences form the basis of HOCC graph generation mechanism. One way of looking at graph generation from a fixed number of records is treating records as vertices and connecting two records by an edge if they share an attribute. The inverse representation is the attribute value pairs from each record forms the vertices in the graph and co-occurrence of two attribute value pairs in a record generates an edges in the graph. Both of these techniques have their advantages and disadvantages and previous work has utilized both these graph representations. HOCC uses the inverse representation. One distinguishing impact

of utilizing co-occurrences is that it violates the Identical and Independent Distribution of instances (or IID) assumption which underlies most statistical techniques. We explain this in the next section.

1.2 A Departure From The IID Assumption

Data has been traditionally treated as Independent and Identically Distributed (IID) [1]. For instance, the roll of fair dice where the result of each roll has no impact on any other. Identical distribution implies that any two significantly large random samples drawn from the data set would have the same underlying distribution. IID is an important assumption underlying several statistical methods and theorems (for instance, the central limit theorem). Statistical techniques typically summarize a distribution to simplify the underlying mathematics and thereby, end up losing valuable information that lies in the order and/or distribution of data [2]. If labeled data is limited and does not capture the underlying model, statistical techniques become less accurate. In general, in the absence of IID data or normality violations, statistical techniques are far less robust. In such cases, it is beneficial to exploit information in relationships between instances [3]. One class of algorithms that utilize such relational information is collective classification.

In general, collective classification utilizes a similarity metric for comparing features of an instance being classified to a set of instances similar to the instance under consideration. Similarity can be defined using various metrics: [11] is based on hyper links whereas in our case we leverage higher-order co-occurrences of items, etc. In more abstract terms, similarity defines the edges in a graph in which instances are the vertices. The location of an instance in the graph is then used to calculate the class of that instance. For example, [11] uses several techniques including a majority operator to label the class of blogs based on the

hyper link structure of the network. In this case the majority operator assigns the label of the majority of its neighbors to a blog.

This thesis presents a novel Higher Order Learning (HOL) based technique. HOL techniques are a new paradigm for statistical machine learning which explicitly violate the IID independence assumption. The techniques are able to extract rich latent information present in higher-order item co-occurrences and outperform first order (i.e., traditional) techniques for supervised classification in the presence of limited training data. The details of the technique are presented in Chapter 3. In the next section, we describe HOL in more detail.

1.3 What Is Higher Order Learning?

Traditional statistical techniques use information (attributes) from each individual data point to build models. The attributes describing each data point are 'first order' information in the sense that they occur in a given data point. In other words, the attributes are not relational in that they are derived from a single instance. As noted, however, [1] concludes that valuable relational information can be exploited in model construction. In fact, when there is little labeled training data, latent information based on higher-order item co-occurrences can be leveraged to build more robust models [4], [5], [6], [7].

A well known algorithm that exploits link structure and thus, violates the IID independence assumption is the Google Page Rank algorithm [8]. It utilizes explicit hyper links to build a graph and then, assigns a weight (authority) to each page based on its position in the graph. Co-occurrence is a widely used concept in everyday life. Consider market basket analysis, where it is likely that customers will purchase spare batteries with a digital camera. Also, it is likely that customers will purchase a digital camera with a Secure Digital (SD) card. Assuming that these transactions have a high frequency but batteries are not

purchased with SD cards, it would be beneficial for a retailer to leverage the higher-order association between the SD card and batteries (through the digital camera) and place these items together on shelves to increase cross-selling. This is an example of a second order association. Such associations are also extensively used in investigations by law enforcement agencies. For example, a large methamphetamine crackdown was executed by the US Drug Enforcement Agency based on higher-order associations in documents, addresses and phone numbers [9]. Literature Based Discovery techniques also leverage such higher-order relations to discover new knowledge. For instance, magnesium was discovered as a possible factor in migraines in a medical informatics study of second-order associations between terms in MEDLINE abstracts [10]. Another interesting ranking metric is the Erdös number. Paul Erdös, a prolific writer and mathematician, was reserved the special author number of zero. Any authors who published a paper with him are assigned a rank of one (a first order co-occurrence). Anyone who publishes with someone who published a paper with Erdös is assigned the number 2 (a second order co-occurrence) and so on. The smaller a person's Erdös number, the better.

In general, any association involving items across records is defined as a higher-order association. The techniques discussed in this thesis rely on the link structure in an item co-occurrence graph, but as noted instead of assigning weights or classifying individual instances, the techniques assign a weight to an entire (sub)graph. The Higher Order Collective Classifier (HOCC) uses such higher-order associations to build robust models with limited data. Our experimental methodologies have generally researched higher-order co-occurrence associations up to order five. As can be seen in Chapter 6, beyond this computation becomes infeasible.

As noted, there have been several learning algorithms developed based on HOL including

Higher Order Naïve Bayes and Higher Order SVM for supervised learning and Distributed Higher Order Association Rule Mining [4], [5], [6], [7]. A general feature of all these algorithms has been the ability to outperform first-order techniques in the presence of very limited training data. HOCC is a novel collective classification technique based on HOL that has been applied to network data.

HOCC has been successfully applied to difficult problems in network security. To the best of my knowledge, it is the only system capable of both anomaly detection and classification of events in Border Gateway Protocol control-plane data. These results are presented in Chapter 4. HOCC is also capable of building models of users from low level Network File System calls in masquerade detection, something that could not be achieved accurately by first-order techniques. These results are presented in Chapter 5. HOCC is applied over a window of time. It leverages the link structure of the instances in the window to build a graph that models the underlying event. From this graph HOCC extracts information to build a distribution of higher-order associations. This distribution can be compared to other distributions to test for statistical similarity. In essence, HOCC provides a measure of difference between two (sub)graphs. As noted we apply this scheme in two completely different problem domains: first, anomaly detection and classification and second, masquerade detection. Though both applications are with network data, the domain can be expanded to include a variety of time-series problems such as detecting the state of activity of the human brain based on EEG data. We discuss HOCC in detail in Chapter 3.

1.4 Thesis Contributions

This thesis makes the following contributions to Higher Order Learning based systems in particular and Computer Science in general:

- A novel technique for extracting higher order features. Previous techniques had larger space and time complexity and hence could not be considered for real time deployment. The new techniques presented in this work can be utilized to extract higher order information efficiently. These techniques can be extended to all previous Higher Order Learning based systems

- A novel higher order learning based collective classifier; HOCC. Collective classification has traditionally worked at the instance level, classifying an instance based on its neighborhood. In this work we present a collective classifier that classifies a collective based on the entire neighborhood

- A novel technique capable of anomaly detection and classification for the BGP dataset. Till recently no single system has been capable of both anomaly detection and classification on the BGP dataset. HOCC is the first system that has shown the potential to do this and that too efficiently.

- A new dataset for masquerade detection.Previous masquerade detection systems relied on a single data set for testing masquerade activity. This has limited the development of masquerade detection systems. The exploration of a completely new dataset for masquerade detection is therefore a significant contribution to the field.

- A novel approach to masquerade detection using HOCC. Given the data, we were able to successfully model user behavior. Using this behavior we were able to differentiate between users with high accuracy.

- Time Complexity improvements in general Higher Order Learning based systems. Previous Higher Order Learning based systems have suffered from very high time complexity. The algorithms for Path Enumeration presented in this thesis were able

to outperform previous algorithms by two orders of magnitude. To give a brief idea, previous techniques would require days to model six minutes of data whereas HOCC, can complete the same task in less than 4 minutes. The algorithms presented can be employed by all higher order learning based systems and improves performance of all HOL systems across the board.

The applications of HOCC presented in this thesis is presently under review at the IEEE Information and Security Informatics conference, 2009.

Chapter 2

Related Work

Modern data mining algorithms primarily leverage first-order relations, aiming to discover patterns such as association or classification rules only within records. The underlying assumption is that records are independent and identically distributed [1]. This assumption simplifies the underlying mathematics of statistical models, but in fact does not hold for many real world applications [2]. Although useful models can be learned, in many cases more accurate models can be learned if higher-order associations are leveraged by the data mining algorithm - associations between records. For example, in supervised learning a classification model is applied to a single record and a prediction is made based on the items (i.e., attribute-value pairs, or features) of the given record in a context-free manner, independent of the other records in the dataset. However, this context-free approach does not exploit the available information about relationships between records in a dataset [3]. Coupled with this is a growing need for data fusion due to increasing data fragmentation [13]. Consequently, there is a greater need to incorporate contextual information - e.g., higher-order associations that cross record boundaries.

Prior work mathematically proved that Latent Semantic Indexing (LSI), a well-known approach to information retrieval, implicitly depends on higher-order co-occurrence associations [5]. This work also demonstrated empirically that higher-order associations play a key

role in the effectiveness of systems based on LSI. LSI can reveal hidden or latent relationships among terms, as terms semantically similar lie closer to each other in the LSI vector space. This can be demonstrated using the LSI term-term co-occurrence matrix. Let's assume a simple document collection where D1 is human, interface and D2 is interface, user. Clearly, the terms "human" and "user" do not co-occur in the co-occurrence matrix of this simple two-document collection. After applying LSI, however, the reduced representation co-occurrence matrix may have a non-zero entry for "human" and "user" thus, implying a similarity between the two terms. This is an example of second order co-occurrence; in other words, there is a second-order path between "human" and "user" through "interface".

In a related effort, [14] uses higher-order co-occurrence to solve a component of the problem of lexical choice, which identifies synonyms in a given context. In another effort, [15] use second order co-occurrence to improve the runtime performance of LSI. Second- and higher-order co-occurrence has also been used in other applications, including word sense disambiguation [16] and stemming [17].

In statistical relational learning (SRL), models operate on relational data that includes explicit links between instances. These relations provide rich information that can be used to improve classification accuracy of learned models since attributes of linked instances are often correlated, and links are more likely to exist between instances that have some commonality [2]. Given a set of test instances, relational models typically simultaneously label all instances in order to exploit the correlations between class labels of related instances. This is called collective classification (or collective inference), and as noted violates the traditional independence assumption. Several studies [18], [19], [1] have shown that by making inferences about multiple data instances simultaneously collective inference can significantly reduce classification error [20].

One of the earliest and best known works in SRL is a probabilistic relational model (PRM) proposed in [22]. This model extends Bayesian networks to the relational domain by incorporating relational information into the dependency structure of traditional Bayesian networks. In follow-on work, [23] uses a structured logistic regression model to capture both content and link information and find that using link distributions improves classification accuracy. One difference of this work from their early work is that they use a discriminative model (logistic regression) rather than a generative model (Naïve Bayes). They use simple link attributes in order to exploit dependencies between related instance labels. In a related effort, [1] presents a form of collective classification in which class labels of all the instances in a test set are simultaneously decided. This allows their technique to explicitly make use of the correlations between the labels of related instances. Their approach leverages Relational Markov Networks, an extension of Markov Networks [25] for relational data. [19] also extends a graphical model, in this case Dependency Networks [21] for relational data. Their technique introduces Relational Dependency Networks, a new form of a PRM that estimates joint probability distributions of relational data. In a recent effort, [3] uses a graphical model that can be considered a first-order Markov random field or Markov network. The authors use an approximation technique termed relaxation labeling for link-based classification. In summary, several recent studies including those mentioned above show that collective classification can significantly improve classification accuracy compared to the traditional classification techniques.

2.1 Prior Work in Higher Order Learning

There have been a number of developments in the field of Higher Order Learning that led to the development of HOCC. In this section we present existing Higher Order Learning based

systems that demonstrate the utility of leveraging higher order co-occurrences or paths.

2.1.1 D-HOTM: Distributed Higher Order Text Mining

D-HOTM is the first system to make explicit use of Higher Order Associations, and lays the foundations of this field [5]. D-HOTM is essentially a higher-order distributed association rule mining algorithm that discovers interesting relationships between items in distributed data. The technique was successfully applied to identify and predict relations between items in e-market basket data. The approach relies on a record-relation graph where each transaction is a vertex in the graph and items that co-occur in different records form the edges. D-HOTM was also applied to predict crime in data obtained from the Richmond, Virginia Police Department. The approach was validated by discovering higher-order associations between criminals and crime types in training data that correctly predicted future criminal behavior [6].

2.1.2 HOPA: Higher Order Path Analysis

HOPA is a Higher Order Learning based approach capable of classifying worm events [4]. Prior work focused on anomaly detection due to the difficulty in distinguishing events, but HOPA succeeded in classifying events. (Additional background and related work is provided in the introduction and related work section of Chapter 4.)

Surprisingly, however, HOPA failed to model an important class: normal. This limits the applicability of the technique since only signatures of known events can be detected. Another significant drawback of HOPA is that it cannot be implemented in real time. With HOPA, building a model of a single snapshot of data on a PC workstation could take a day or more depending on the density and number of vertices in the graph.

HOCC is a direct successor of HOPA. At an abstract level, the goals of the two systems are essentially the same, but the techniques used for graph comparison differ. Unlike HOPA, the novel comparison technique introduced in HOCC is capable of defining a model of normal behavior. This implies HOCC can not only distinguish between various events but also identify anomalies. As noted, HOCC also has significantly less time complexity, building models in minutes instead of hours.

2.1.3 HONB: Higher Order Naïve Bayes

HONB is a supervised Higher Order Learning algorithm [24]. As the name suggests, the approach is based on Naïve Bayes. Instead of utilizing priors based on term frequencies in documents, HONB employs priors derived from the higher-order path counts. The approach constructs an item relation graph for each class and calculates higher-order priors for each item in each class. HONB was validated on a number of datasets including the 20 Newsgroups, a widely accepted standard in text classification. The performance in the presence of limited training data compares favorably to the well known Support Vector Machine, outperforming SVM by a significant margin. In the experiments reported in [24], classification performance improved by 27% over Naïve Bayes and 14% over SVM.

In summary, previous work in Higher Order Learning has demonstrated the importance of utilizing information present in higher-order co-occurrences, or paths [4], [5], [6], [7], [24].

The work in this thesis falls into three broad categories: first, the HOCC system itself and how it leverages higher order associations in modeling; second, applications of HOCC in the cyber-security domain in chapters 4 and 5; and third, improvements to run-time performance of HOCC in chapter 6. In the following chapter, we present HOCC.

Chapter 3

HOCC: A Higher Order Collective Classifier

In this chapter, we present the theory underlying HOCC. We define the item co-occurrence graph, paths and record sets followed by record set enumeration. To set the context, we first present an overview of the operation of HOCC: HOCC creates a graph from instances aggregated from a window. Each window has a fixed number of instances. Each instance consists of a set of attribute-value pairs, or items. The vertices of the graph are unique items, while the edges are generated based on co-occurrence of items in one or more instances. Each edge in the graph is labeled by the identifier(s) of the instance(s) in which the two items co-occur. As such, in time series data the window and resulting graph represent a snapshot of ongoing activity. Graphs are characterized using various techniques, such as item set enumeration [4] or record set enumeration, the latter of which is described in this thesis. The characterization of a given window of a known event is termed a model of that event.

HOCC compares signatures of known models (e.g., of worms or users) to models of real-time network data. An event is either classified or an anomaly or masquerade is detected when the real-time model is statistically significantly different from the known model.

With this context in mind, the following section presents the theory behind HOCC's graph characterization and comparison technique. Note that in what follows a 'record' is identical to an 'instance.'

3.1 Theory

Most of the definitions presented in this section can be found in graph theory text books such as [27], [26]. The theorems and definitions provide the basis of HOCC and also the logic behind the design decisions. We start by defining a window.

Definition 3.1.1 *A window R is a set of records $r_i \in R$ with fixed cardinality. Each record r_i is a set of unique (attribute, value) pairs, also referred to as items.*

This definition for a window can be extended because a value could imply an integer, floating point number, string or an object such as an image, audio or video file. For the data sets that we employed in the experimental results reported in Chapters 4 and 5, however, the values were exclusively integers. Another constraint not explicit in this definition is that we applied HOCC to time series data. A time constraint is not included in the definition, however, because HOCC need not be constrained to time series data alone.

There is a trade-off involved with the window cardinality (size) and time taken to analyze a given window. We will later see that linearly increasing the window size exponentially increases processing time. Also, the degree of overlap between successive windows is limited by the available compute power. As a result, the key parameters for an HOCC model is cardinality of a window or window size and the bin size. We use the term bin because in most of the applications we aggregated data over a small time span and binned it together to form a single record. For the purposes of this discussion, the terms bin and record should be treated as equivalent.

Definition 3.1.2 *An item co-occurrence graph $G = (V, E)$ is an undirected multi-graph, where V is the set of vertices and E is the set of edges. Given a dataset R, vertex set V is the set of all distinct attribute-value pairs of R. An edge $e_{(u,v,i)}$ connects two vertices u*

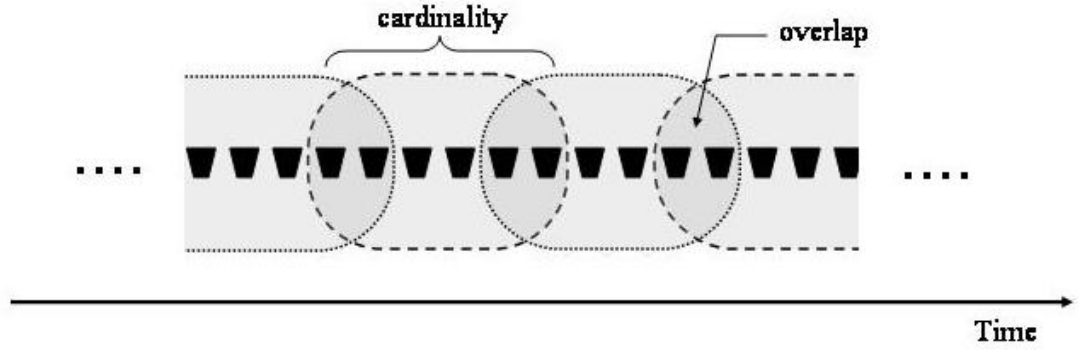

Figure 3.1: Windowing Over A Time Series Of Bins. Each Bin Is Equivalent To A Record.

and v if and only if there exists a record $r_i \in R$ that contains both attribute-value pairs u and v.

Since G is undirected, if an edge $e_{u,v,i}$ exists, then $e_{v,u,i}$ also exists and both represent the same edge. Each edge is also labeled by r_i, the record that contains both items. Two attribute-value pairs can belong to two different records. This means it is possible to have multiple edges between pairs of vertices in the graph. This is why the co-occurrence graph is a multi-graph. Also, by Definitions 3.1.1 and 3.1.2, each record $r_i \in R$ can be represented as a fixed subset of the vertices V from graph G.

The number of vertices in the graph is equal to the number of unique attribute-value pairs in a window. This is a number that lies between the number of attributes (if all attributes of a given type have the exact same value) and the number of attributes multiplied by the number of records in a window (when each attribute-value pair is unique). Generally, as the window size increases linearly, the number of vertices (attribute-value pairs) in the graph increases linearly. In Chapter 6, we will see the effect on HOCC's time complexity as the number of vertices increases linearly.

A simple path in the graph is a sequence of connected non-repeated vertices. The length of any simple path in G is bounded by $|V|$-1. If there exists a simple path P consisting of $\{v_i$ where $i \in [1, p]\}$, then there must exist a reverse path P' consisting of $\{v_j$ where $j \in [p, 1]\}$

The terms forward and reverse paths in this context should not be confused with the same terms used for directed graphs. For example, a forward path is of the form $x - y - z$, where x, y and z are attribute-value pairs, then the reverse of this path is $z - y - x$. Both of these are distinct paths in the undirected graph G and are mirror images of each other.

A direct corollary of the above statements is :

Corollary 3.1.3 *The number of forward paths is equal to the number of reverse paths in G.*

3.2 Record Set Enumeration

We turn now to record set enumeration, which is the key to the novelty of HOCC. We start by defining a record set.

Definition 3.2.1 *A record set is an ordered collection of non repeated records $ri \in R$ in an item co-occurrence graph $G = (V, E)$ such that there exists a simple path P consisting of v_i where $i \in 1, p$ such that each $r_j \in e_{j,j+1}$ where $j \in (1, p-1)$. By definition the size of a record set is the length of the path.*

Figure 3.2 depicts a simple example. The definition ensures that record boundaries are crossed at least once to create a record set. Counting record sets is significantly faster than the bipartite matching technique used to count item sets in [4], as we can quickly compute all record sets by simply taking the cross product of all sets of records in a given path.

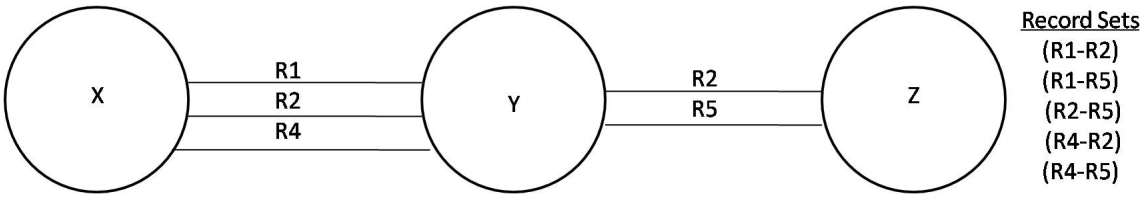

Figure 3.2: Example: Record Set Enumeration for $p = 2$. X,Y and Z represent attribute-value pairs in an Item Co-occurrence graph. The records connecting two items are the records where they co-occur. The record sets generated from these attribute value pairs are represented on the right.

Using the ordering information from a path distinguishes it from the technique used in [6]. The definition also guarantees the exclusive use of higher order co-occurrence information to generate the distribution of the record set frequencies. The record set frequencies are utilized to create the frequent-frequent record set distribution described in the next section.

Definition 3.2.2 *A higher order path is an ordered set of vertices than lie along a simple path of length greater than one with no two edges labeled with the same record.*

This means every higher-order path crosses record boundaries. Simple paths of length one do not qualify as a higher order path because they imply first order co-occurrence. A direct corollary of the above two definitions is that each record set belongs to some higher order path in the graph.

In order to enumerate all record sets, HOCC enumerates all paths in the graph and then for each path calculates all record sets. Each distinct record set occurs one or more times. HOCC uses the frequency of occurrence of the distinct record sets to calculate what we term the frequent-frequent record set distribution. Before discussing record set enumeration further, however, we give a short proof that leads to a minor optimization.

Lemma 3.2.3 *If A is a multi set representing the frequency of distinct record sets for an item co-occurrence graph G and F represents the frequency of distinct record sets obtained*

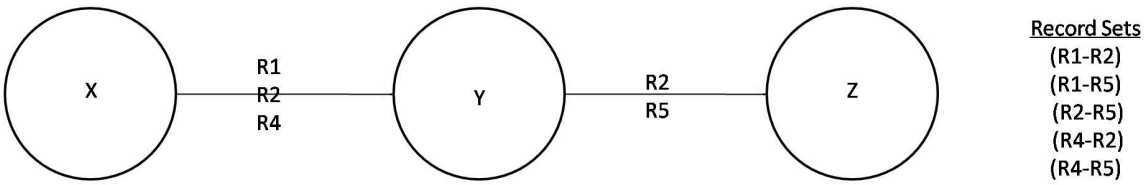

Figure 3.3: Condensed version of the graph in Figure 3.2

from only forward path in a graph, then $A = F \cup F$.

Proof: A can be represented as the multi set representing the frequencies of distinct record sets for all forward (F) and reverse (R) paths. Therefore, $A = F \cup R$. Each path at most generates one record set. The record sets generated from the reverse paths are an exact mirror of the record sets generated from the forward paths by Definition 3.2.1. For example, if a forward path generates a single record set $r_x; r_y; r_z$ then the reverse will generate $r_z; r_y; r_x$. By Corollary 3.1.3, the number of distinct forward paths is equal to the counts of the reverse paths. Therefore, the counts of the distinct record sets paths generated by a forward path will be the same as the counts of record sets generated by the reverse path. In other words, $F = R$. Replacing R by F, we get $A = F \cup F$

Each path containing unique labeled edges qualifies as a higher order path and the labels of the edges taken in order is a record set for the path. Thus, in the multi-graph, each simple path generates at most a single record set.

As far as implementation is concerned, it is too expensive to enumerate all paths in a multi-graph. Therefore, all edges in the item set co-occurrence graph between a pair of vertices are represented as a single edge. This is the condensed version of the multi-graph. Each edge is represented by a set of record labels. Each label implies a co-occurrence between two items in a record r_i. Figure 3.3 is the condensed version of Figure 3.2. Each path thus corresponds to an ordered, non-empty set of records. HOCC chooses a record

from each edge (set) such that no two records are repeated. This satisfies the definition of a record set and guarantees that higher order associations are leveraged.

In the worst case, the record set enumeration cost is $O(|R|^{|P|})$, where $|R|$ is the window cardinality and $|P|$ is the path length. But, in practice, co-occurence of items is not so high and record set enumeration completes in unit time. This worst case complexity would only arise if all edges were between the same pair of vertices which means, in the condensed version, the number of vertices is going to be small and path enumeration will complete quickly. Path enumeration is the bottleneck in HOCC, and hence, time complexity of the record set enumeration can be ignored.

3.3 Frequent-Frequent Record Set Distribution

Once all record sets have been enumerated, the frequency of each unique record set in all higher order paths in the graph is counted. This is denoted as the frequency of a record set. HOCC then enumerates the number of record sets that share the same frequency. This distribution is denoted as the frequent-frequent or f-f record set count and represents the underlying model of an event. HOCC uses this distribution to compare against other models during anomaly or masquerade detection, as well as during event classification.

The number of record sets in a graph is bounded by $\binom{O(}{VP)}$, where V is the number of vertices in a graph and P is the length of each path. Each record set is denoted by x_i. HOCC counts the frequency of each record set generated from all paths in the graph and denotes this by $c(x_i)$. This set can be large and as noted HOCC reduces this further by calculating the frequency of the frequency of each record set, $c(c(x_i))$. Figure 3.4 depicts a simple example. This frequent-frequent record set distribution represents the model of the underlying event. We compare event models by comparing these distributions using the

Record Set	Frequency
R1,R4,R6	1
R4,R5,R1	2
R2,R6,R5	4
R3,R5,R1	4
R1,R3,R4	1
R3,R4,R5	4
R4,R5,R6	8

Frequency	Frequent – Frequent Record Set
1	2
2	1
4	3
8	1

Figure 3.4: Frequent-Frequent Record Set Enumeration

Students T-Test.

In this manner, HOCC generates a distribution of integers from a window based on the link structure of the graph. The information extracted for building the distribution is local in nature in that it reflects the influence of near neighbors of items in the co-occurrence graph - i.e., it depends on higher order paths. However, instead of using this information directly to classify instances, HOCC globally aggregates the higher order relations to generate the frequent-frequent record set distribution, thereby collectively modeling all instances in the graph. This is a key distinction of HOCC from standard collective classification techniques.

In conclusion, Figure 3.5 portrays the entire process in a generic application of HOCC to network data.

In this chapter, we have described HOCC in detail, with the sole exception of the path enumeration algorithm. We present this in Chapter 6 with proofs of correctness, asymptotic time complexity analyses and a direct comparison with previous Higher Order Learning systems.

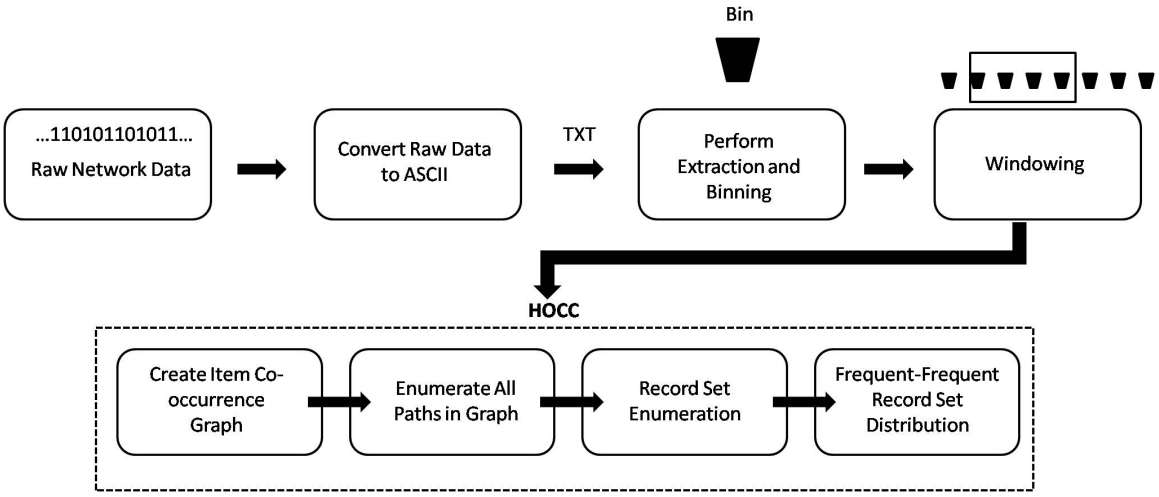

Figure 3.5: The General Framework Of HOCC Applied To Network Data

Chapter 4

Anomaly Detection & Classification: The BGP Dataset

4.1 Introduction

Border Gateway Protocol (BGP) is the *de facto* inter domain routing protocol. BGP is responsible for the discovery and maintenance of paths between autonomous systems (ASes) in the Internet. The Internet is made of thousands of ASes, which are loosely defined as a connected set of IP prefixes under a single administration [28]. BGP provides reachability information to ASes and distributes external reachability internally within an AS. With the exponential growth of ASes, BGP has become one of the most critical components of the Internet's infrastructure. Within the last few years, Internet routing dynamics have been extensively studied [29], [30], [31], [32].

The Internet has become the backbone of much import, and attacks on BGP servers have the potential to make systems and services unreachable. This is unacceptable given the dependence on the Internet for so many aspects of modern-day life. Events such as the East Coast electricity blackout of August 2003 brought down hundreds of BGP servers, completely disrupting the routing fabric. Misconfiguration of BGP servers can also disrupt routing [33]. Finally, the BGP routing fabric can and has been severely affected by worms. Each of these events - natural phenomena, malicious worms and mis configurations - have different impacts on the Internet, and it is thus important to precisely identify the nature of such anomalous events.

In fact, a precise diagnosis is important because different events require different corrective action. For instance, some events could cause a surge of traffic but at little to no cost to the network [34]. Others might require rebuilding of routing tables [35], and worms require quarantine/containment techniques. An accurate and timely diagnosis of the anomalous events is thus necessary for determining the right corrective action to aid a quick recovery.

The Slammer (also known as Sapphire) worm infected machines running Microsoft SQL Server starting January 25th 2003. It was one the fastest spreading worms of the time. The only limiting factor to the spread of Slammer were other Slammer-infected machines competing for the same resources. Slammer took several database servers out of action. Access to several websites was restricted because of bandwidth saturation. Fortunately, for the Internet, Slammer did not carry a malicious payload. Had that been the case, the consequences could have been very severe. Even though the Slammer worm only affected SQL servers, its impact was observable in traces of the BGP control-plane traffic. The primary reason Slammer could not be contained was because it spread far too quickly for detection and human intervention. Quick identification of the spread of the epidemic would have helped in shutting down services or quarantining infected machines thereby reducing damage [36].

In contrast to Slammer, the Witty worm infected only a fraction of the number of servers Slammer infected. But what it lacked in numbers, it compensated for in malice. Once the worm infected a machine, it sent copies of itself to 20,000 random destinations. Each packet was of a random but small size, making it difficult to filter and small enough that it would not be broken down or readily dropped by networks. The worm then randomly over-wrote 65 Kilobytes of data on the hard drive of infected machines, and repeated this until the machine was brought down. The worm spread very rapidly and within 45 minutes

had infected 12,000 machines. The only reason for the relatively small number of infected machines was the fact that infected machines would completely stop or reboot, killing the worm and curtailing further spread of infection. Similar to Slammer, Witty could not be contained due to the rapid rate at which it spread, leaving insufficient time for detection and human intervention [37].

The Blackout event on the other hand was caused by the loss of electricity to 3175 BGP servers on August 14, 2003. The event resulted in loss of connectivity to several institutions and individuals for hours to days. This event required regeneration of paths in the network. Although the event was not necessarily of a malicious nature, its impact was very similar. It resulted in frenetic activity on the part of the adjacent BGP servers because known paths had disappeared. However, the impact of the event was local in nature and did not result in any cascaded failures elsewhere [38].

4.2 BGP Anomaly Detection and Event Classification

[39] provides an excellent survey of general anomaly detection problems and techniques. There have been several successful systems developed for the detection of anomalies in BGP traffic. Some previous techniques applied to BGP data include using neural networks and rule mining techniques that can detect novelty (or anomalies), such as those presented in [40] and [41] respectively. In [41], Li et al. use attributes derived from BGP traffic to detect Internet routing anomalies. They employ data mining techniques, in particular a decision tree machine learning algorithm, to train a model using labeled data. The authors use the counts of different types of BGP messages divided into one minute bins. Their model consists of the rules learned, and is used to detect occurrences of abnormal events. Basically, their system can distinguish between two classes - event and normal - but cannot

identify the exact type of an event. Thus, one important drawback in their approach is that it cannot distinguish between different anomalous events such as worms. In fact, in her public review, Dina Katabi from MIT points out the importance of identifying whether an abnormal event is caused by a worm, blackout or misconfiguration [41]. Similarly , [40] achieves success in detecting novel events in BGP but is unable to differentiate between them.

Several other research efforts of a similar nature have been conducted in [28], [17], [15] and [42]. [15] proposes two approaches, signature-based and statistics-based detection. The authors employ wavelets and k-means clustering to build an instance-learning framework that identifies anomalies for a given prefix as well as across prefixes. Most of these efforts follow the same basic steps: first, the system is trained using labeled training data, and then the system examines test data and flags anomalies. In summary, previous work has been focused on the detection of anomalies rather than the classification of events, and is thus in general unable to differentiate between events.

One significant exception that succeeded in the classification of events is [4] which is able to distinguish between events with very high accuracy. However, the authors were unable to build a model for non-anomalous events (normal behavior). As a result, the approach cannot be used for anomaly detection *per se*. In addition, the approach has a high time complexity which prevents it from being used in practice (this point is discussed in Chapter 6). This then forms the basis for our work. We build on the work completed in [4] and develop a novel approach capable of precisely classifying events *as well as* identifying anomalies. More significantly, we achieve this in real-time based on a novel approach that leverages our research in Higher Order Learning.

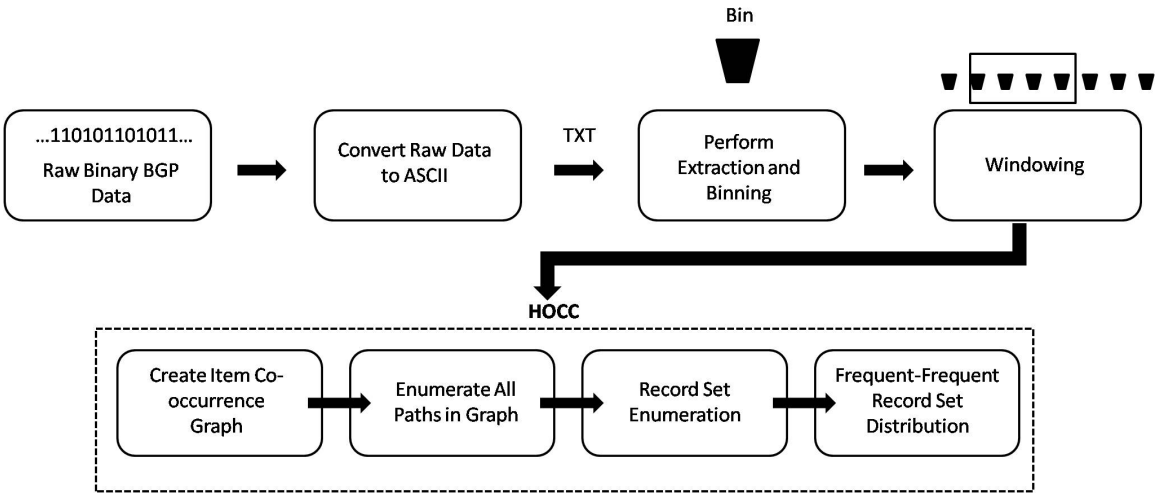

Figure 4.1: BGP Anomaly Detection System

4.3 Results: Application of HOCC to BGP Data

Figure 4.1. shows an overview of the system for real time anomaly detection and classification.

The BGP control plane data is in binary. We use free tools provided by [43] to convert the binary files into plain text representing packet level information. This information is continuous and each packet has a time stamp. We use these time stamps to create three second bins. Each three second bin consists of a variable number of packets. We extract the following information for each three second bin:

1. Number of Announcements
2. Number of Withdrawals
3. Number of Announcements and Withdrawals
4. Number of AS prefixes announced. Each announcement (1) can announce several AS prefixes
5. Number of AS prefix withdrawals. Each withdrawal (2) can withdraw several AS

prefixes

6. Number of AS prefix announcements and withdrawals

The bins are generated every three seconds. We create a sliding window over 120 of such three second bins. HOCC is applied to each of these sliding windows. Each sliding window is defined as a model of the event. We define three models that we shall refer to frequently.

- **Real Time Model:** The 120 three second bins window that is obtained from real time BGP event data is the real time model.
- **Normal Model:** This model is a window over 120 (labeled) normal bins. The normal model is used in detection of anomalies.
- **Event Model:** This model is again a window of 120 (labeled) event bins. This model represents the signature of the previously known event and is used for classification.

To perform anomaly detection, we compare the Real Time Model to the Normal Model while for classification we compare the Real Time Model to the Event Models of all previously known events. The models consist of a single window containing 120 three second bins. We compare models by performing the student's T-test between the frequent-frequent record set distributions of the two events generated by HOCC. The results indicate that we can successfully model an event using a single window (1/480 of the available data).

For Witty and Slammer, a data set of 600 bins, 300 of which are pre-event and 300 post was employed, while for Blackout a dataset of 800 bins, 400 of which are pre-event and 400 post was employed. As noted, we created a sliding window containing 120 three second bins. For each window, we constructed the item co-occurrence graph and listed paths of length three ($|P| = 3$). 480 windows (600 - 120 bins) were created for Slammer and Witty and 680 for Blackout, with each window as noted consisting of 120 bins each.

Table 4.1: Event vs Event comparison

Event 1	Event 2	T-Test result
Slammer	Witty	0.00016127
Blackout	Witty	0.031218
Slammer	Blackout	0.036645
Normal	Slammer	1.06×10^{-5}
Normal	Witty	0.035918647
Normal	Blackout	0.000401

The results are depicted in *Figures* 4.2, 4.5, 4.3, 4.6, 4.4 and 4.7 below. In these figures, the values along the X-axis are the window numbers while the Y axis corresponds to the value of the t-test ranging between 0 and 1. The red line at the bottom is a measure of significance set to 5%. The thin green line indicates the point in time where the bins from the event start filtering into the window and the thick green line indicates the point at which the sliding window is completely into the event.

4.3.1 Supervised Classification

As discussed previously, HOCC statistically distinguishes between events based on t-tests between event models. The results in Table 4.1 indicate that the event models are statistically significantly different based on t-test p-values. Our results support the conclusion that these events can be distinctly identified and distinguished.

Figures 4.2, 4.3 and 4.4 portray the p-values of the t-test comparisons between the real time models (sliding windows across time) and the event models of Witty, Slammer and Blackout respectively. In Witty and Slammer, HOCC detected a peak almost exactly at the point the window starts moving into the event. For Blackout, the event is detected about 10 windows or 30 seconds later. Larger p-values of the T-test indicate that the distribution of the real time model matches the distribution of the event model. Small values indicate

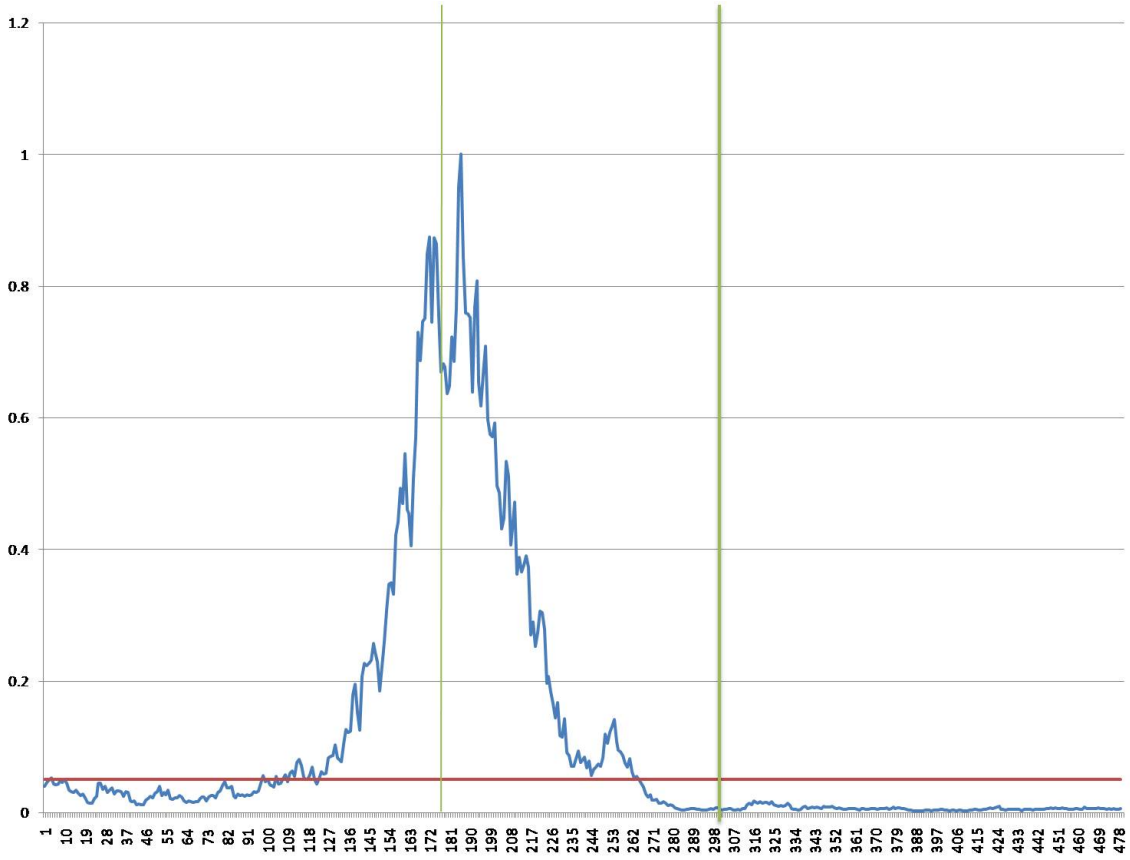

Figure 4.2: Witty Supervised

that the models are dissimilar. The results of the t-test p-values remain consistently greater than 5% for Slammer and Blackout once the windows move into the event and less than 5% otherwise. For Witty, the t-test p-values rise sharply and significantly at the start of event but immediately after, decline sharply. We speculate that this is due to the aforementioned nature of Witty - it was not widespread due to the destructive payload, and previous work has largely ignored it because it is so difficult to model.

4.3.2 Anomaly/Novelty Detection

In this section, we present results of anomaly detection in which we compared the real time event model with the normal event model. Figures 4.5, 4.6 and 4.7 show results of anomaly

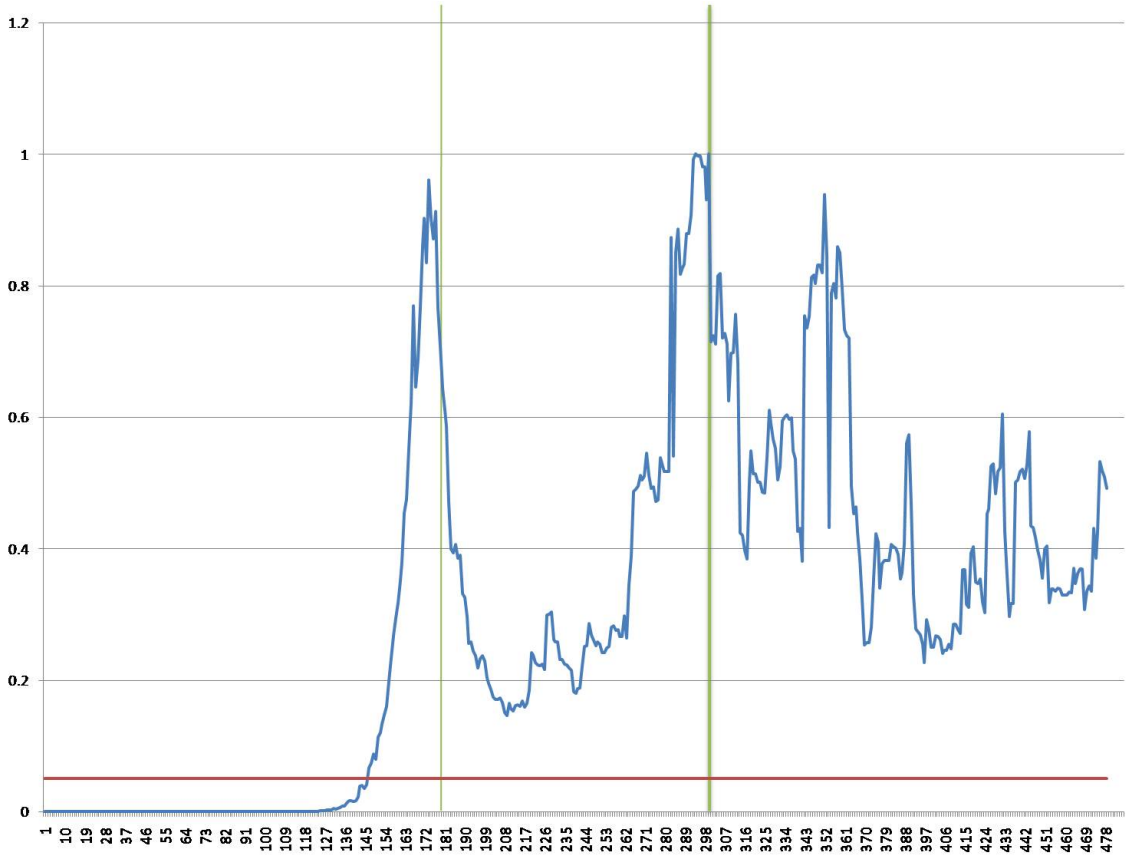

Figure 4.3: Slammer Supervised

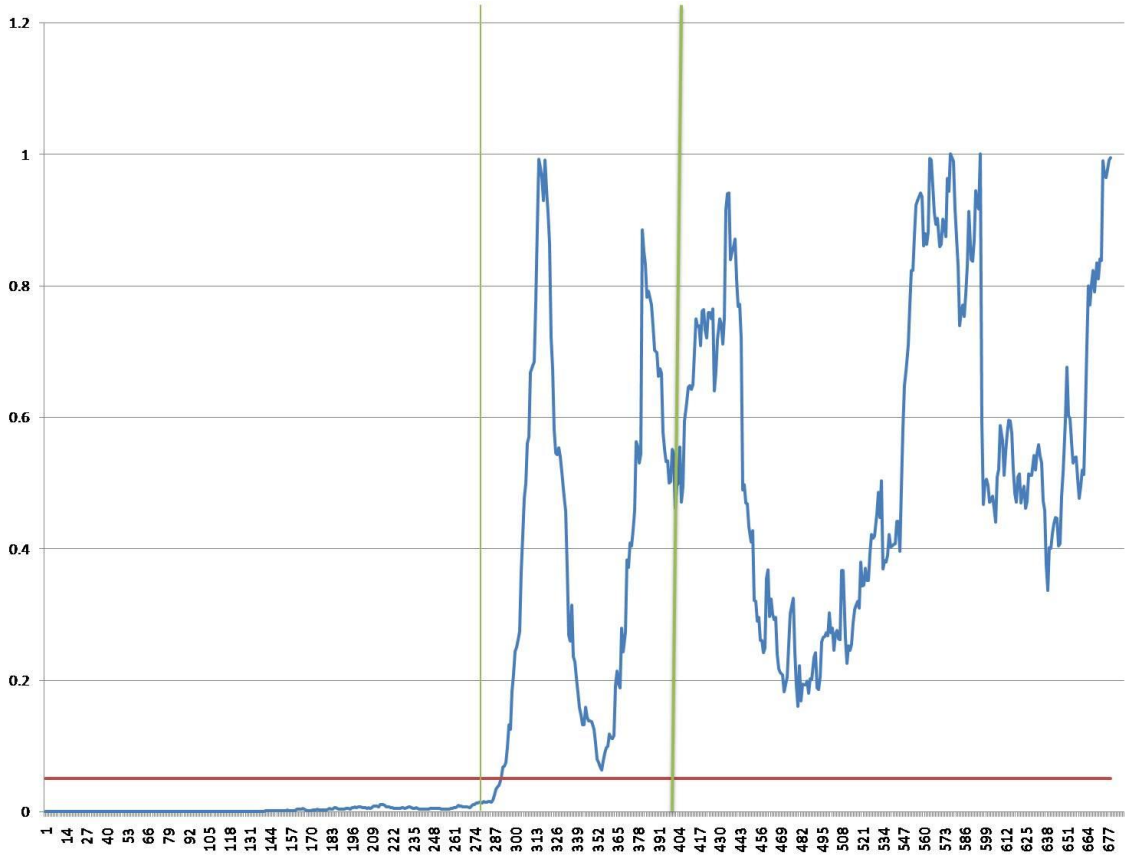

Figure 4.4: Blackout Supervised

detection using HOCC to identify the Witty, Slammer and Blackout events as anomalies, not events per se. Here, the t-test values were obtained by comparing the real time model to the normal model. We observe large t-test p-values (closer to 1) before the events begin. This is consistent with the fact that the pre-event period is normal, and the t-test values do not lead us to conclude that the models are statistically significantly different. As the windows move into the anomaly, however, the t-test values fall significantly (to less than 5%). This drop is very large for Slammer and Blackout where the t-test values are of the order of magnitude 10^{-5} and 10^{-4} respectively. For Witty, the drop is not as large, but enough to be statistically significant. This indicates that the anomalies are statistically significantly different from normal. The t-test p-values remain significant for Slammer and Blackout, but return to almost normal for Witty. This behavior is consistent with our earlier results when classifying Witty - in essence, to the best of our knowledge HOCC is the first learning algorithm capable of modeling Witty for event classification as well as for anomaly detection.

4.4 Discussion

As we have already mentioned in Section 4.2: Related work, anomaly detection and classification in the BGP control plane has been a difficult problem. This is primarily due to the unavailability of training data. Data for BGP anomaly detection and classification is sparse because anomalous events not only occur rarely but also rarely if ever recur. This means that there is very little training data for rule mining or classification algorithms to use and build robust systems. Traditional rule mining and classification techniques end up over fitting the data, making event classification difficult. Then, what is it that makes Higher Order Learning techniques work so consistently in such situations? The answer lies in the

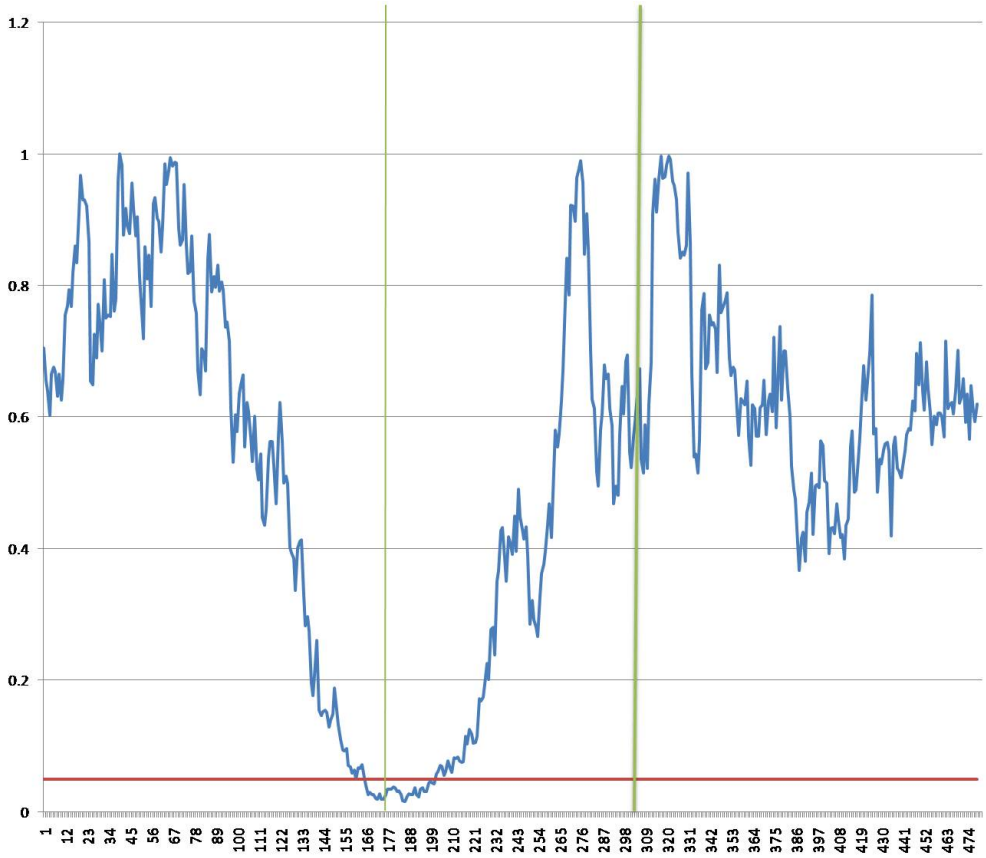

Figure 4.5: Witty Unsupervised

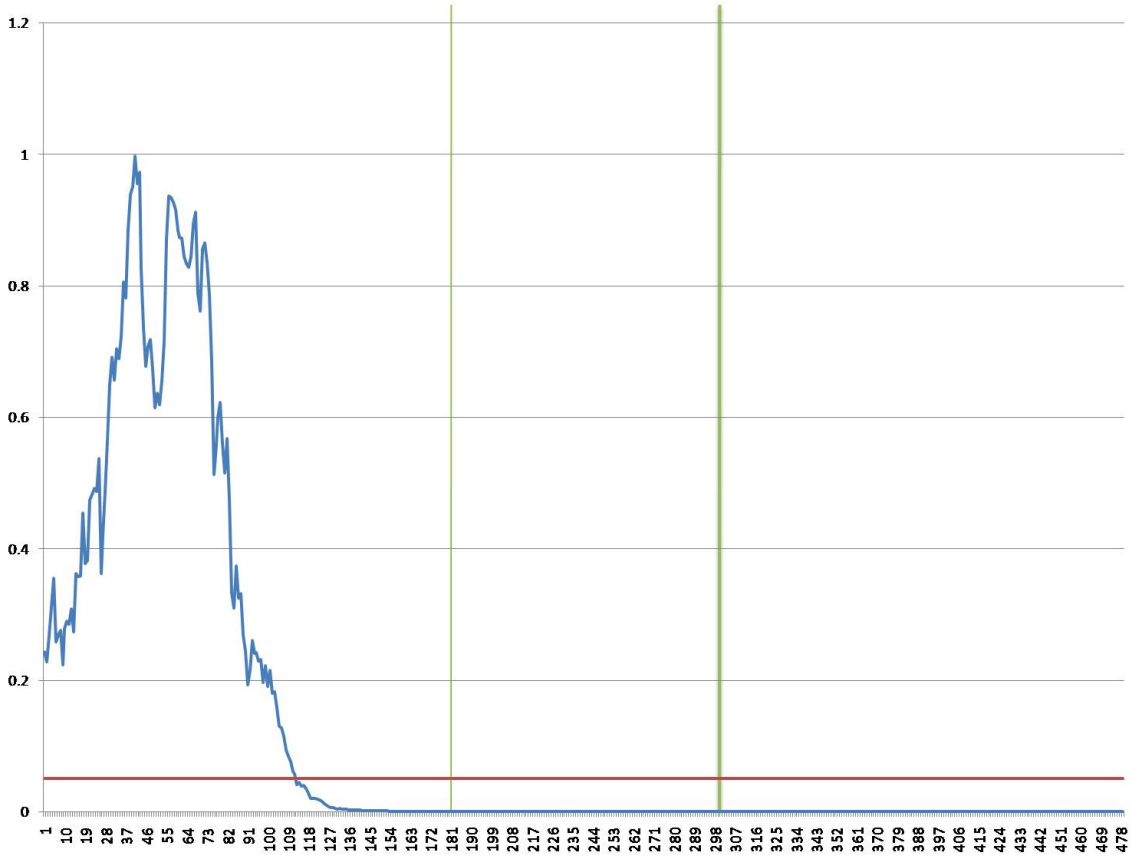

Figure 4.6: Slammer Unsupervised

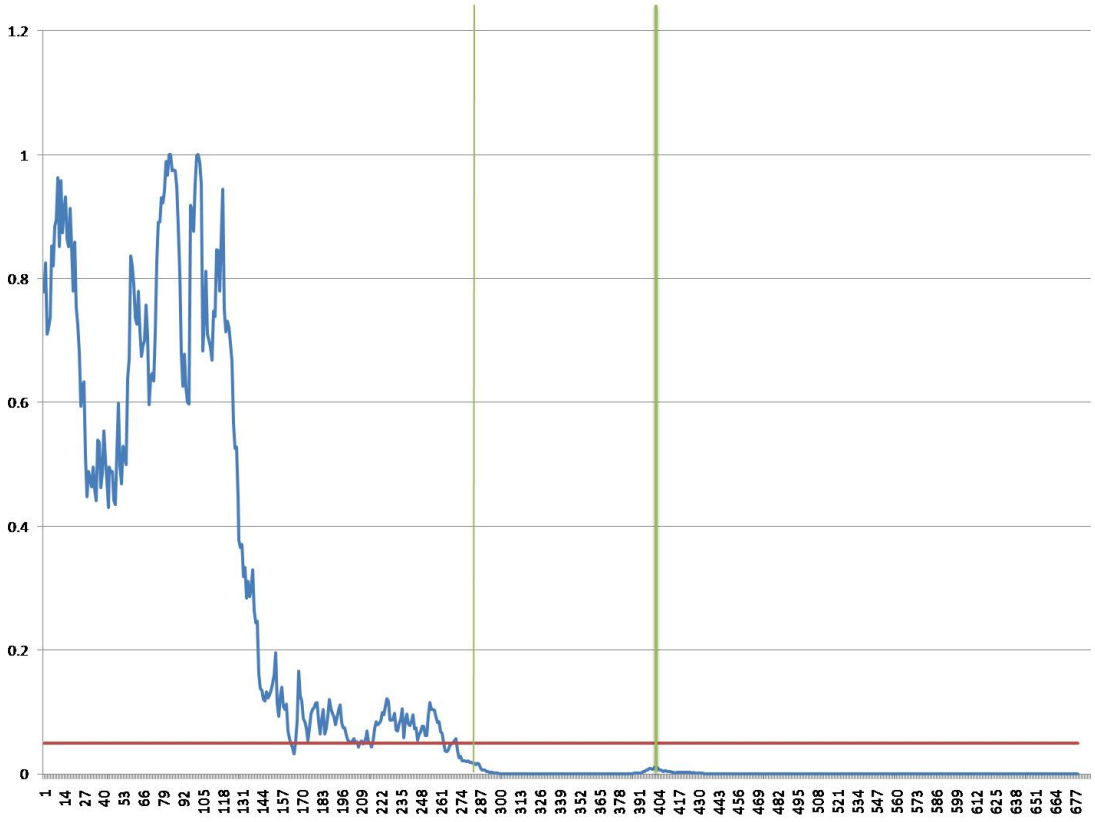

Figure 4.7: Blackout Unsupervised

ability of Higher Order Learning algorithms to construct robust models using very little training data. This has been a feature of Higher Order Learning techniques right from the start. For instance, the Higher Order Naïve Bayes algorithm has been shown to outperform standard Naïve Bayes, as well as state of the art SVM classification algorithms for text classification in the presence of very small training sets. As expected of course, increasing the training data improves performance of first-order (non-higher order) algorithms but in some cases Higher Order Learning algorithms continue to exceed the performance of their first-order analogues even as samples scale to 90% of the training data (the usual percentage used in 10-fold cross validation). We believe this is the reason that HOCC has proven capable of doing what no other first-order based technique has accomplished: given just 120 instances of an event, HOCC builds robust models useful in both classification and anomaly detection.

Chapter 5

Masquerade Detection: The Network File System Dataset

5.1 Introduction

We use the Harvard Network File System(NFS) dataset for masquerade detection [56]. The purpose for collecting such a large corpus was to analyze and compare the trace with similar prior traces, analyze performance and impact on file servers and analyze the correlations between attributes in the dataset. This data set was not intended for use in a security context. We explored how this data set could be used for masquerade detection. To the best of my knowledge, no other work has been reported in the literature that has attempted using the Harvard NFS dataset for masquerade detection on this scale.

The Network File System (NFS) Dataset consists of network file system accesses at Harvard University for two groups, the Electrical Engineering and Computer Science graduate students and CAMPUS the central computing facility at the university. For privacy reasons, the dataset has been anonymized. The anonymization process replaced all UID (User Identity), GID (Group Identity) and IP (Internet Protocol) addresses with arbitrary but consistent values. The data set contains a wealth of information and can be 'replayed' in real time to simulate the entire traffic stream.

Because of the nature of the anonymization, we make a few assumptions. One important underlying assumption is that the data contains no masquerades. This way, we can assume that each user was him/herself and test for masquerades by comparing one user model with

another. Another assumption is that although the users' behavior changes over time, the change is of such a nature that the core behavior of a user remains most like him/herself. This assumption is important and enables us frame the solution as a maximum likelihood problem. Our experimental results confirm the validity of this latter assumption. These results are also presented in [44]. In the following sections, we explain the use of HOCC for detection of masquerades and compare results with first order systems on this NFS dataset.

5.2 Masquerade Detection

A masquerader impersonates another user to obtain privileges not available to him/her. These privileges can be used to either access unauthorized information from the system (a security breach) or perform a malicious attack on the system that could either damage it or convert the system into a host machine for launching future attacks. Sophisticated masqueraders generally possess information about the system, network topologies, system vulnerabilities and the exact nature of installed security systems/mechanisms/processes. Trivial access control mechanisms cannot stop such masqueraders because they possess adequate information to bypass them. In fact, more often than not, the masqueraders are authentic system users who have both knowledge and expertise to fool the system into believing they are another user.

Masquerade detection has been a difficult problem. To make matters worse, data for masquerade detection is difficult to obtain primarily due to privacy concerns [45], [46], [47], [48]. A problem that makes masquerade detection difficult is the fact that user behavior changes over time, so a user's behavior today can potentially be completely different a year later. False positives have plagued several masquerade detection systems. In addition, data collection efforts have not focused on security. This has made quantifying performance

across different datasets difficult. However, using the Harvard NFS traces - a new, large corpus of labeled data - it should be possible to address these problems.

In the next section, we discuss how HOCC is applied to the Harvard NFS dataset.

5.2.1 The Dataset

We chose to model the EECS dataset consisting mostly of *research* users. This data set should be more difficult to model because of the extensive use of scripts and daemons by advanced users. The data set is characterized by a higher ratio of writes to reads against the CAMPUS data set, which predominantly consists of reads (over 95%). The EECS data set is smaller and consists of only 283 million calls (approximately 120 GB uncompressed) while the CAMPUS dataset is larger and occupies 320 GB when compressed.

The dataset is divided into two sets, one of NFS calls and another of NFS replies. The calls are made by users to the NFS server and the replies are the NFS server's responses to the calls. It is important to note that users do not explicitly make calls to the NFS server; rather, calls for service are made by numerous applications on the users' behalf. In this sense, the calls represent a user's pattern of use of applications and utilities that access the NFS. As such, they are intuitively a good basis for modeling user activity. As a result, we chose to filter replies and model only calls. Each call is composed of several attributes, which were down-selected to a small subset, described in the next section.

5.2.2 Attribute Selection

The call dataset consists of 40 attributes, many of which are undefined depending on the executing operation. From these 40, we selected a subset of seven attributes. We choose these attributes based on the application of attribute subset selection techniques in the

WEKA Workbench [50]. For attribute selection, we employed two WEKA methods, Decision Trees and Linear Support Vector Machines. We first applied the WEKA J48 Decision Tree learner to determine attributes that were not selected by the algorithm when classifying user ID. We then executed the Linear SVM and ranked attributes with their coefficients. If an attribute was not selected and had a small co-efficient, it was dropped. This process was repeated until we converged on the seven final attributes. The final set of attributes selected is: time stamp, source address, source port, user ID, group ID, call type (which could be one of 21 different NFS call types), Acc (Access permission type) and Stable_how (this is a bit that indicates the nature of a write).

It is important to note that domain expertise also argues for including these particular attributes. For instance, it is a well known fact that IP addresses and ports, the nature of calls made, etc. are important to model a user's behavior. Related to this, one interesting attribute throughout this work has been GID. A masquerader is likely going to get the necessary group privileges of the user he is emulating, which means incorporating GID may be of little value. Modeling without GID proved difficult for the first-order techniques, however. As a result, we performed experiments both with and without GID to evaluate first-order techniques but all the results for HOCC were generated without the GID attribute.

5.2.3 Methodology & Results

We selected 10 users randomly from the NFS file traces. For each of these 10 users we choose a random start point in time from the traces and collected 500,000 raw NFS calls. From these calls, we extracted information from the seven selected attributes. These calls were then divided into two categories, one for training and a second for testing. The initial calls in the sequence were used for training and the remaining for testing. The time span

for these calls was approximately two weeks.

Using the time stamps, we binned the data into 2, 4, 8, 16, 32, 64, 128 and 256 second intervals. Each bin consisted of the following information; total number of calls, total number of source addresses, total number of source ports, total number of calls of each of the 21 types of calls, total number of groups, total number of acc and total number of stable_how. Thus, we created eight series of binned data.

For each of the bin series, windows were created ranging in size of 10, 20, 40, 80, 120 and 160 bins. Each of these windows was used to generate the item co-occurrence graph that models user activity for that period. Note that in this case an 'item' is defined as a particular value of one of the seven attributes. HOCC was used for the comparison of these models. As noted in the Introduction to this chapter, we expected the learned models of each user to conform with the test models for the same user. Using every possible combination of bin series and window size, we constructed 10 models for each user. We present the results obtained for models with two second bins and a window size of 80 in Table 5.1.

In Table 5.1, the columns correspond to the models learned for each user and the rows correspond to models of other users used for testing. Each comparison is the average p-value of 200 comparisons (10 training models $\times$ 20 test models). For example, the first value in the table corresponds to the comparison of column 18a88 training set versus row user 18a88 test set. A large value reflects greater similarity. To be precise, although anything larger than a value of p=0.05 would be sufficient to conclude that two distributions are not statistically significantly different, a larger p-value indicates lesser dissimilarity. The ideal comparison table would then be a unit matrix with ones along the diagonal and zeros off diagonal. However, as noted there is variation even between models learned for the same user. Also, there is undoubtedly noise due to background daemons and services. This makes

Table 5.1: User vs. User comparison. Two Second Bins With Window Size 80

User	18a88	18a8a	18a89	18a8e	18a9f	18aa9	18aaa	18ad2	18ad6	18b17
18a88	**0.9535**	0.3811	0.1618	0.0556	0.0623	0.0031	0.1891	0.0072	0.8206	0.2367
18a8a	0.2666	**0.7349**	0.1648	0.1464	0.0003	5.19E-07	0.2425	3.06E-06	0.3757	0.0056
18a89	0.1616	0.1637	**0.9339**	0.1741	0.1582	0.1563	0.1908	0.1568	0.1621	0.1593
18a8e	0.0576	0.1036	0.1769	**0.8357**	0.0146	0.0068	0.5584	0.0083	0.0685	0.0227
18a9f	0.0840	0.0008	0.1585	0.0114	**0.7974**	1.31E-06	0.1382	0.0006	0.0607	0.1659
18aa9	0.0054	8.73E-07	0.1565	0.0044	4.15E-06	**0.6855**	0.1143	0.0001	0.0043	2.00E-07
18aaa	0.1868	0.2230	0.1928	0.4781	0.1357	0.1133	**0.9543**	0.1188	0.1953	0.1513
18ad2	0.0109	6.11E-06	0.1570	0.0055	0.0008	8.05E-06	0.11953	**0.5410**	0.0083	2.05E-05
18ad6	0.8560	0.4896	0.1621	0.0663	0.0911	0.0081	0.1941	0.0159	**0.8608**	0.2616
18b17	0.2206	0.0087	0.1594	0.0174	0.1806	1.40E-07	0.1508	2.37E-05	0.1621	**0.7312**

user modeling a difficult problem, in particular for first order techniques that end up over fitting and in the worst case, performing at the baseline.

As can be seen in Table 5.1, our results show large values along the diagonal with an average of 0.8 whereas the non-diagonal elements average 0.13.

One of the concerns with this technique is that some users may have small p-values when compared to themselves, e.g., for user 18ad2 the p-value is 0.541. In other cases, during testing we might find a test user that is not significantly different from the user model under consideration. To resolve these issues, we employ a maximum likelihood framework for one class classification [44]. In such cases, we apply the MAX operator on the corresponding row. The MAX operator sets the largest average p-value to one and zero to the rest of the values in the row. In case there are two equally large, MAX randomly chooses one. In the case of user 18ad2, this maximum likelihood framework tells us that the user indeed is most like himself (the largest values are highlighted in bold). This can also be confirmed by applying the MAX operator to each column. For this simple MAX operator, both precision and recall are defined as the ratio of the sum of the diagonal to the total number of users (since there can only exist one maximum in each row and the rest are zero). As a result, for this small set of users we achieved both 100% recall and 100% precision. A notable instance is user 18ad6. This user, even though most similar to himself, is also quite similar to user 18a88. Even though this instance satisfies the MAX operator property and is correctly identified, it is expected that a good technique should strongly differentiate between the two. For this reason, we also tried comparing users with different models. For example, using 64 second bins and windows of size 80 we can strongly distinguish between user 18ad6 and 18a88 as shown in Table 5.2. Thus, a second approach optimized for a particular user could be used for validation.

Table 5.2: User vs. User comparison. 64 Second Bins With Window Size 80

User	18a88	18ad6
18a88	**0.7455**	0.1308
18ad6	0.1437	**0.9028**

When we choose the best models for each user for comparison, the diagonal has a larger average p-value of 0.89 and the average of non-diagonal elements falls to 0.11. This is depicted in Table 5.3 By building custom models for each user in the maximum likelihood framework, we again achieved both 100% recall and 100% precision for this set of users, similar to the results in Table 5.1. In both cases, by applying the MAX operator we were able to achived ideal performance with 100% precision and 100% recall using just 160 seconds of labeled data to build our models.

In complete contrast to these results, first-order techniques achieved only low accuracies. We explored the parameter space in experiments with first-order algorithms by varying the dataset used (raw data vs. the binned, aggregated data), as well as by conducting experiments both with and without GID. We first ran the Nave Bayes algorithm on the raw dataset without binning. The class for each instance was set to UID and the classifier was trained using 10 fold cross validation, implying 90% of the data was used to train the models. Even though we gave it substantially more data to learn from, including the GID attribute, the Nave Bayes classifier achieved an accuracy of only 70.07%. Also, as the number of users increased (i.e., number of classes increased) the quality of models dropped significantly. For instance, Nave Bayes achieved a classification accuracy of only 50% on a 20 user dataset. We also used the more sophisticated J48 Decision Tree learner for identifying users. With GID, although 10-fold CV accuracy was high, J48 built a tree of depth 125 with 105 leaves that seriously overfit on GID. After removing the GID attribute, the J48 performance dropped to 70% accuracy.

Table 5.3: User vs. User comparison. Largest p-value Per User

User	18a88	18a8a	18a89	18a8e	18a9f	18aa9	18aaa	18ad2	18ad6	18b17
18a88	**0.9535**	0.3811	0.1618	0.0556	0.0623	0.0031	0.1891	0.0072	0.8206	0.2368
18a8a	0.2546	**0.9302**	0.4010	0.0135	0.0771	0.11037	0.00890	2.76E-07	0.7702	0.0246
18a89	0.1886	0.1935	**0.9594**	0.2476	0.1896	1.90E-03	0.1917	0.1877	0.1927	0.1910
18a8e	0.0322	0.0426	0.2849	**0.9418**	0.0584	0.0459	0.6347	0.0289	0.7760	0.0336
18a9f	0.1513	0.0549	0.1747	0.0128	**0.9033**	6.79E-12	0.0517	2.22E-09	0.2454	0.6841
18aa9	0.0014	0.0002	0.1401	0.0358	3.93E-12	**0.75419**	0.1541	0.1971	0.2203	2.80E-10
18aaa	0.0328	0.8347	0.2358	0.0100	0.1345	0.21	**0.9597**	0.0202	0.5250	0.3643
18ad2	0.0117	4.41E-08	0.1420	0.0019	4.45E-05	0.145561	0.0189	**0.7581**	0.0860	0.0009
18ad6	0.1438	0.8369	0.1887	0.0113	0.2674	2.30E-03	0.7134	0.0755	**0.9027**	0.5572
18b17	0.0152	0.0127	0.1448	0.0636	0.0922	1.27E-10	0.1614	4.46E-09	0.2517	**0.7955**

It may be argued that HOCC had an advantage due to using binned, aggregated data. To level the playing field, we also evaluated the first-order learners using the same binned data provided to HOCC. In this case, Nave Bayes' performance averaged 37% for the 10 users. J48 was able to do slightly better with an average accuracy of 47%. These first-order learner results are clearly unacceptable for use in masquerade detection systems.

5.3 Discussion and Concluding Remarks

The results indicate that HOCC can successfully build models of users and can distinguish between them. This is a significant step forward from first-order systems which suffered from unacceptably low accuracy and high false positive rates. For HOCC, even the simplistic MAX operator was sufficient to provide exceptionally good performance in terms of accuracy, precision and recall. As the number of users increases, however, it is likely more users could be similar to each other. As demonstrated in our experimental results, in such cases it is possible to select model parameters (bin and window size) that are better at discriminating between users rather than using a single set of parameters for comparison.

These masquerade detection techniques can be integrated with FileWall [49] policies. FileWall is a network middlebox that enables the use of context-aware policies for file systems using both network and file system context. FileWall has been designed to minimize administrative overheads for common file system operation and has been tested under heavy loads. Such integration is left for future work, but the primary issue to be addressed is the complexity of path enumeration. We address this issue further in the following chapter, Chapter 6. Another interesting area of possible future work is the use of approximation techniques for estimating the frequent-frequent record set distributions. This is a promising direction, and has potential for use in other Higher Order Learning algorithms as well.

Finally, record set enumeration is amenable to parallelization - an increasingly important direction given the trend towards more and more cores in standard off-the-shelf processors. We address this point further in Chapter 6 as well.

One limitation of HOCC is that the approach depends on obtaining at least one window of NFS calls for the model. A masquerader who has knowledge of this approach, however, could attack by making few calls in a short burst with a frequency designed to avoid detection. The burst could potentially affect only a single bin in any given window. Such a scheme might not generate a graph sufficiently distinct from the user's. For this reason, an approach that detects intrusions in a small sample is needed. In other words, from a security perspective it is advantageous to choose smaller windows so that the percentage of masquerade calls to user calls within a given window is high. In contrast, our results indicate that HOCC is more accurate for larger windows. This issue will likely be difficult to resolve and will involve trade-offs in security, accuracy or time.

A highlight of this research is the fact that in it we have leveraged a new dataset for security using a novel approach to collective classification based on Higher Order Learning. The ability to build user models from this dataset indicates that HOCC can be successfully employed to detect masquerades. Prior to this, efforts reported in the literature have relied exclusively on a single dataset for masquerade detection, the Schonlau data set [48]. Given the unavailability of data for security applications, the NFS dataset provides a large quantity of high quality data for testing of security systems.

Chapter 6

Time Complexity Optimizations for Higher Order Learning Based Systems

All Higher Order Learning techniques are based on utilizing path information present in a record-relation or entity-relation graph. Thus, a common theme in Higher Order Learning is the need to perform path enumeration. In particular, it is necessary to enumerate all paths in a graph up to a fixed length.

As noted, Higher Order Learning techniques are based on the idea of discovering all possible connections between two vertices in a graph. This is reflected in the algorithms that were utilized in first generation Higher Order Learning approaches, which attempted to solve this all-pairs / all-paths problem [4], [6], [7]. These approaches are based on iteration over all pairs, which is $O(n^2)$, followed by the use of [51]'s algorithm for enumeration of all paths between a given pair of vertices. The complexity of path enumeration for each path using [51] in the worst case is $O(VE)$ where V is the number of vertices and E is the number of edges.

Several graph theory techniques [27], [26] were explored in prior work including transitive closure and the Floyd-Warshall closure which provide reachability information between two vertices but do not provide the exact paths between the two. Modifying these algorithms to generate all paths requires complex data structures which add to the time complexity. Powers of the adjacency matrix can also be used for determining counts of paths. For

example, the square of an adjacency matrix identifies the unique second-order paths, but not the exact number of paths connecting any given pair of vertices.

In [52], it was shown that all paths in a graph could be enumerated in $O(V^3)$ matrix operations. However, this bound relied on the ability to perform matrix operations in constant time, which is infeasible on a general purpose architecture. There do however exist approximate algorithms that provide the counts of the number of paths in the graph such as [53], [54]. But these algorithms do not provide the precise path counts that have been employed so successfully in Higher Order Learning approaches to date. Nonetheless, in the future work the efficacy of approximate algorithms will be explored in the context of Higher Order Learning. For now, however, the challenge is to enumerate paths efficiently to support real-time applications.

Generation of All paths in a Graph

```
1: procedure GENERATE ALL PATHS(G =(V,E))
2:     p ← Desired Path Length.
3:     for v such that v ∈ V do
4:         P ← φ.
5:         P ← v.
6:         Process Paths (P, AdjList(v) ,p)
7:     end for
8: end procedure
```

Process Paths

```
1: function PROCESS PATHS(P,V′,p)
2:     if Length(P) < p then
3:         Print path P
4:         for v ∈ V′ do
5:             if v ∉ Path P then
6:                 P′ ← P + v
7:                 Process path (P′, AdjList(v), p)
8:             end if
9:         end for
10:    end if
11: end function
```

6.1 Proof of Correctness:

Algorithm "*Generation of All paths in a Graph*" generates all valid paths of length one. *Process path* adds each element from the adjacency list of the source vertex one at a time to generate new paths. The adjacency list of the source vertex corresponds to paths of length one. This is repeated for each vertex in the graph. Thus, each edge or all paths of length one get generated.

Assuming the algorithm generates all paths of length n, we prove by induction that all paths of length n + 1 are also generated. There exists some path of length n from source v_1 to v_n. v_{n+1} is some vertex in the adjacency list of v_n.

There are three possible reasons why a path from v_1 to v_{n+1} would not be generated.

1. v_{n+1} already lies along the path P. But then, by definition v_1 to v_{n+1} would contain repeated vertices and would not qualify as a path.

2. v_{n+1} does not lie along the adjacency list of v_n. But then, no path starting from v_1 through v_n to v_{n+1} of length n+1 could exist.

3. The path length between v_1 and v_{n+1} must be greater than n+1. We know that v_1 to v_n is a path of length n. Therefore, v_n to v_{n+1} must be a path of length greater than one. But v_{n+1} lies in the adjacency list of v_n and has a distance of one. Therefore, v_1 to v_{n+1} must be of length n+1.

Because none of the above is possible, if a path exists, it will be discovered. It can be similarly argued to prove that the algorithm does not generate non-existent paths.

The next question that needs to be answered is, "Does the algorithm terminate?". This

is answered by calculating the asymptotic complexity of the algorithm, the topic of the next section.

6.2 Asymptotic Complexity Analysis

Every call to *Process path* creates a path of length less than P. The output step in the *Process path* procedure takes $O(P)$ time where P is the path length. The depth of this recursion is the path length, and is $O(V^P)$. Each call takes $O(P)$ time giving us a total complexity of $O(P.V^P)$. It could also be argued that the output step takes constant time, but that is not true in this case because the algorithm traverses a linked list to output the path. It also could be argued that using strings would make this step take constant or $O(1)$ time. However, the use of strings would require insertion and deletion at the end of the strings which would require $O(P)$ time. Hence, we conclude that the output step in *Process path* takes $O(P)$ time.

The insert procedure traverses the adjacency list, calling *Process path* $O(V)$ times.

Each call to *Process path* generates one new path. If the total number of paths is $O(VP)$, the overall time taken is $O(V^P.P)$. In contrast, with the previous technique, each path takes worst case O(V.E) time giving us a total complexity of $O(V^{P+1}.E)$. Since path length is generally some fixed constant, the new approach is $O(V.E)$ faster than the previous approach. As we shall see in the next section, practically speaking this has a significant impact on the run-time of path enumeration.

6.3 Empirical Results

We present some empirical results obtained by comparing small and sparse graphs using the path enumeration approach employed by [4] (HOPA) and the approach discussed in this

thesis. The graphs were randomly generated using [55].

Figure 6.1 represents the time taken by [4] verses HOCC to perform path enumeration for graphs of low density (0.1). The Y axis represents the log of the run-time and the X axis represents the number of vertices in the graph. Each curve represents either HOPA or HOCC. The numbers alongside represent the path lengths. For instance "HOCC 3" is path enumeration using the technique employed by HOCC for paths of length three. It is worth noting that HOCC is capable of enumerating paths of length six faster than HOPA can enumerate paths of length five. For larger and denser graphs, this performance gap is even greater.

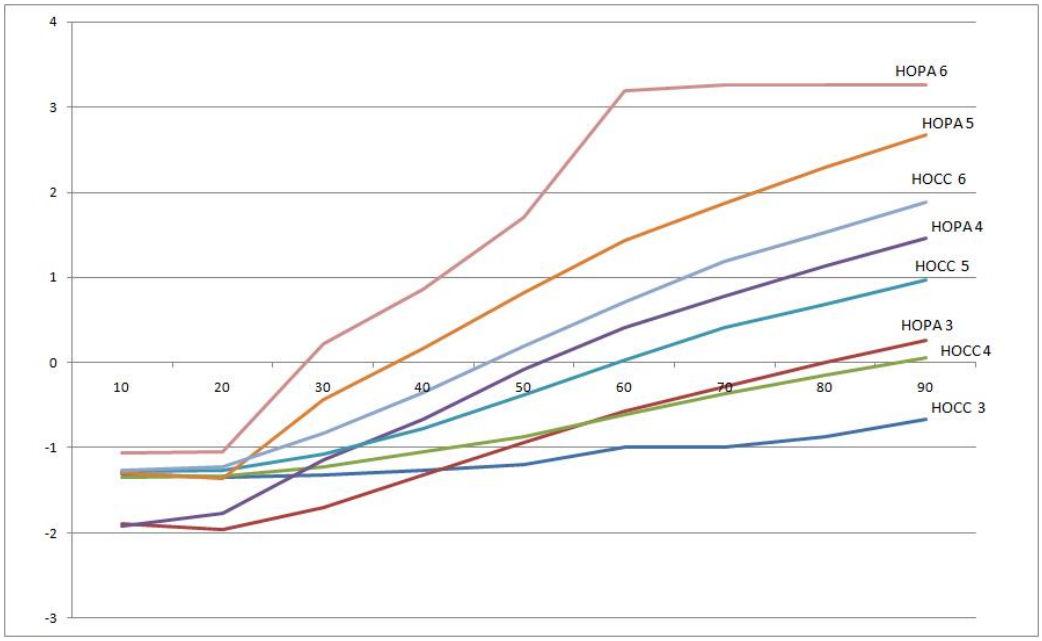

Figure 6.1: Comparison of HOCC to HOPA

Next, to give an idea of the exponential nature of the path enumeration problem we enumerate all paths using both HOCC and HOPA in graphs of density 0.1 and 0.2 in *Figures* 6.2 and 6.3 respectively.

Finally, as a side note, the graphs generated from the BGP dataset consisted of an

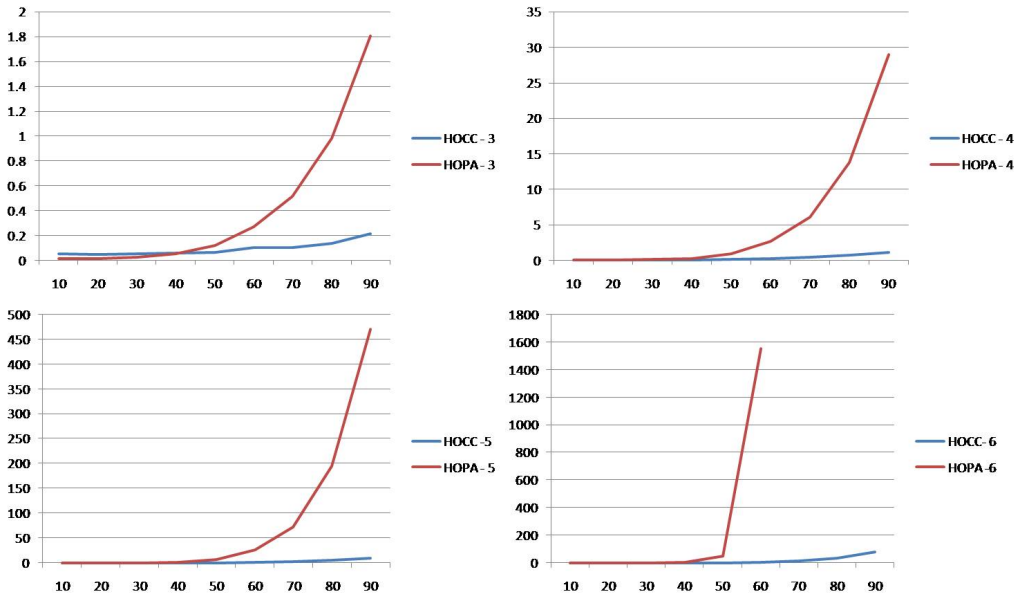

Figure 6.2: HOCC vs HOPA: Graph Density 0.1. X Axis Is The Number Of Vertices In The Graph And Y Axis Is Time In Seconds

average of 500 vertices with edge density ranging from 0.16 to 0.35. HOPA in some cases took over twenty hours for a single six-minute window to complete. In comparison, HOCC modeled each six minute window in less than five minutes.

6.4 Leveraging Multi-Core Technologies

Although the present algorithm is essentially sequential in nature, it is possible to parallelize this algorithm. In one such parallel implementation, each processing unit could make a copy of the graph under consideration along with a set of vertices on which it would operate. Assuming the respective sets correspond to the different classes of the model, the computation time on each individual processing unit would be proportional to number of vertices in the given set. Of course, it is somewhat naïve to assume this, and if it is not the case then interprocessor communication will be needed to enumerate paths between adjacent vertices that belong to different sets (and are therefore on different processing units). Under more or less ideal conditions in which the sets are independent (i.e., belong to different classes)

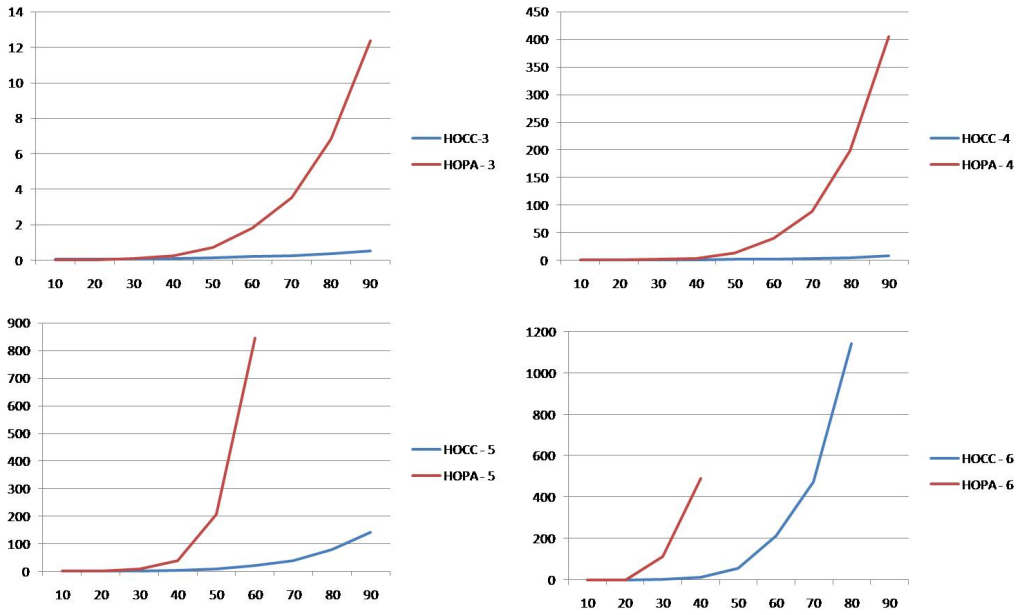

Figure 6.3: HOCC vs HOPA: Graph Density 0.2. X Axis Is The Number Of Vertices In The Graph And Y Axis Is Time In Seconds

and are also of similar size and density, the speedup will be proportional to the number of processing units. In this case, the *Process path* procedure remains unchanged.

Task Divider

1: **procedure** TASK DIVIDER(G =(V,E), N)
2: Divide the set of vertices into K equal partitions where each partition size is V/N
3: **for** each V' such that $V' \in K$ **do**
4: Generate All Paths (G =(V,E), V')
5: **end for**
6: **end procedure**

In the BGP dataset experiments conducted for this thesis, this level of performance was not necessary. However, parallelization could prove quite useful subject to the validity of the assumptions discussed above.

Generation of All paths in a Graph: Parallel Implementation

```
1: procedure GENERATE ALL PATHS(G =(V,E), V')
2:     p ← Desired Path Length.
3:     for v such that v ∈ V' do
4:         P ← φ.
5:         P ← v.
6:         Process Paths (P, AdjList(v) ,p)
7:     end for
8: end procedure
```

Chapter 7

Conclusions and Future Work

In this thesis, we have presented a novel collective classification technique based on Higher Order Learning, HOCC. The learning algorithm has been applied to two difficult problems in network security; anomaly detection and classification and masquerade detection. The performance of HOCC is favorable compared to first-order techniques. Also, the time complexity of HOCC is significantly reduced from that of previous Higher Order Learning algorithms. Higher Order Learning has consistently performed well in the presence of a very limited labeled data. This has time and again been demonstrated in prior work including HOPA [4] and HONB [24] . HOCC also has now demonstrated this same point with the new NFS dataset. In fact, a related contribution of this work has been the use of the NFS dataset in masquerade detection, an especially important application given the scarcity of high quality data for security related problems.

One of the most important aspects of future work lies in improving the complexity of path enumeration, which is in complexity class #P. Although HOCC operated sufficiently efficiently for use in real-time on the BGP dataset, it was unable to achieve real-time performance on the NFS data. Several suggestions were made of possible directions for future work including approximation algorithms, parallelization, etc. Given that path enumeration is in #P, a polynomial time approximation should be realizable. A related issue not

addressed in this thesis is the space complexity, which is also exponential if paths are actually stored. In HOCC, this issue has been mitigated by performing record-set enumeration on the fly, meaning that as soon as a new path is discovered, HOCC extracts all possible record sets from the path. Nevertheless, a polynomial amount of space $O(|W||P|)$ where $|W|$ is the window size and $|P|$ is the path length) is required to store all record sets.

Although HOCC was successful at identifying and classifying network events, another important next step is prescribing a diagnosis. HOCC is purely a statistical tool and has no expertise on the underlying data. Therefore, suggesting a prescription poses a challenge. However, what HOCC can do is provide a prescription when it matches the signature of a known event, if available.

Finally, we close with an intriguing question: How scalable is HOCC in terms of domain? In the experiments conducted in this thesis, we dealt specifically with integer data. However, prior work in Higher Order Leaning dealt with both textual data and e-market basket data. Scaling HOCC to other data types could prove quite interesting. From another perspective, in some ways HOCC is akin to Hidden Markov Models (HMMs) in the sense that in an HMM a probable path is predicted based on known transition probabilities. Could a Higher Order HMM be constructed that incorporates transition probabilities from multiple routes between a pair of given nodes? It would also be interesting to compare HMMs with HOCC in terms of performance because HMMs are extremely fast. The MAX operator defined for use with the Harvard NFS data set is also similar in some ways to the maximum likelihood framework used in HMMs. Based on this, my intuition is that HOCC can be scaled to the same application domains that currently employ HMMs.

References

[1] Taskar, B., Abbeel, P., Koller, D. *Discriminative Probabilistic Models for Relational Data.* Association for Uncertainity in Artificial Intelligence. pg: 485-492. 2002.

[2] Getoor, L., Diehl, C. P. *Introduction to the special issue on link mining.* Association for Computing Machinery (ACM) Special Interest Group on Knowledge Discovery and Data Mining 7(2):1-2. 2005.

[3] Angelova, R., Weikum, G. *Graph-based text classification: learn from your neighbors.* Special Interest Group on Information Retrieval. pg:485-492. 2006.

[4] Ganiz, M.C., Kanitkar, S.,Chuah, M.C., Pottenger, W.M.*Detection of Inter domain Routing Anomalies Based on Higher-Order Path Analysis.* Proceedings of the Sixth International Conference on Data Mining (ICDM) 2006.

[5] Kontostathis, A., Pottenger, W. M. *A Framework for Understanding LSI Performance*. Information Processing & Management, Volume 42, Issue 1, Pages 56-73. 2006.

[6] Li, S., Wu, T., Pottenger, W. M. *Distributed higher order association rule mining using information extracted from textual* data. Special Interest Group on Knowledge Discpveru and Data Mining. Volume 7(1): pg 26-35. 2005.

[7] Li, S., Janneck, C. D., Belapurkar, A. P., Ganiz, M. C., Yang, X., Dilsizian, M., Wu, T., Bright, J. M. , Pottenger, W. M. *Mining Higher-Order Association Rules from Distributed Named Entity Databases.* IEEE Information and Security Informatics. pg: 236-243. 2007.

[8] Page, L., Brin, S., Motwani, R., Winograd, T. *The PageRank Citation Ranking: Bringing Order to the Web.* 1999.

[9] Krauss, C. *U.S. Moves to Close Canadian Drug Route for Illegal Stimulant.* The New York Times, March 5, 2002.

[10] Swanson, D.R., Smalheiser N.R., Torvik V.I.*Ranking Indirect Connections in Literature-Based Discovery: The Role of Medical Subject Headings*. Journal of the American Society for Information Science and Technology, Volume 57(11), pg: 14271439. 2006.

[11] Bhagat, S., Cormode, G., Rozenbaum, I.*Applying Link-based Classification to Label Blogs.* Special Interest Group on Knowledge Discovery and Data Mining Workshop on Web Mining and Web Usage Analysis (WEBKDD) pg: 97:117. 2007.

[12] Nikolov, A., Pottenger, W.M.*Privacy preserving higher-order association rule mining* IEEE Workshop on Information and Security Informatics. 2009.

[13] Rahm, E., Bernstein, P. A. *A survey of approaches to automatic schema matching.* Very Large Data Bases Journal. (VLDB) Volume 10(4), pg:334-350. 2001.

[14] Edmonds, P. *Choosing the word most typical in context using a lexical co-occurrence network.* In Proceedings of the Thirty-fifth Annual Meeting of the Association for Computational Linguistics, pg: 507-509. 1997.

[15] Zhang, X., Berry, M., Raghavan, P.*Level search schemes for information filtering and retrieval.* Information Processing and Management Volume 37 (2), pg: 313-334. 2000.

[16] Schutze, H. *Automatic word sense discrimination.* Computational Linguistics, Volume 24(1), pg:97-124, 1998.

[17] Xu, J., Croft, W. B. *Corpus-based stemming using co-occurrence of word variants.* ACM Transactions on Information Systems Volume 16 (1), pg: 61-81. 1998.

[18] Chakrabarti, S., Dom, B., P. Indyk. *Enhanced Hypertext Classification Using Hyper-Links.* In Proceedings of ACM Special Interest Group on Management of Data Conference, pg. 307-318. 1998.

[19] Neville J., Jensen, D. *Dependency Networks for Relational Data.* IEEE International Conference on Data Mining pg:170-177. 2004.

[20] Jensen, D., Neville, J., Gallagher, B. *Why collective inference improves relational classification.* Association for Computing Machinery's Knowledge Discovery and Data Mining pg:593-598. 2004.

[21] Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., Kadie, C. M. *Dependency Networks for Inference, Collaborative Filtering, and Data Visualization.* Journal of Machine Learning Research (JMLR) Volume 1, pg: 49-75. 2000.

[22] Getoor, L., Friedman, N., Koller, D., Taskar, B. *Learning Probabilistic Relational Models.* Eighteenth International Conference on Machine Learning (ICML), Williams College. June 2001.

[23] Lu, Q., Getoor, L. *Link-based classification.* In Proceedings of the Twentieth International Conference on Machine Learning (ICML), pg: 496-503. 2003.

[24] M. C. Ganiz, N. I. Lytkin, W. M. Pottenger. *A General Approach to Text Classification: Leveraging Higher Order Dependencies Between Features in Small Training Samples.* The 26th International Conference on Machine Learning. 2009.

[25] Pearl, J.*Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufman Publishers. 1988.

[26] Cormen, T. H., Leiserson, C. E., Rivest R. L., Stein, C. *Introduction to Algorithms, Second Edition* ISBN 81-203-2141-3, pg: 581-700.

[27] Knuth, D. E. *The Art of Computer Programming Volume 1. Fundamental Algorithms, Third Edition* ISBN 0-201-89685-0 pg: 309-434.

[28] Zhao, X., Pei, D., Wang, L., Massey, D., Mankin, A., Wu, S., and Zhang, L.*Detection of Invalid Routing Announcements in the Internet.* Proceedings of Dependable Systems and Networks. 2002.

[29] Giffin, T. *What is the Sound of One Route Flapping?* Institute for Pure and Applied Mathematics. 2002.

[30] Caesar,M. , Subramanian, L. , Katz, R.H. *Route Cause analysis of Interdomain routing Dynamics*. Technical Report, University of California Berkeley CSD-04-1302, 2003.

[31] Lad, M., Nanavati, A. and Massey, D. *An Algorithmic Approach to identifying Link Failures.* Proceedings of Pacific Rim Dependable Computing Symposium , March 2004.

[32] Mau, Z.M., Bush, R., Griffin, T.G. and Roughan, M. *BGP Beacons.* Proceedings of Association for Computing Machinery Internet Measurement Conference, 2003.

[33] Mahajan, R., Wetherall, D., and Anderson, T. *Understanding BGP Misconfiguration.* Proceedings of Association for Computing Machinery Special Interest Group on Data Communications, Aug 2002.

[34] Roughan, M., Li, J., Bush, R., Mao, Z., Griffin, T.,*Is BGP update storm a sign of trouble: observing the Internet control and data planes during Internet worms.* Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems, 2006.

[35] Kuhn, D.R.,Sriram, K., Montgomery, D.*Border Gateway Protocol Security.* NIST Special Publication 800-54 (Guidance Document for the Telecom Industry and US Government agencies), 2007.

[36] Moore, D., Paxson, V., Savage, S., Shannon, C.,Staniford, S.,Weaver, N. *Inside the Slammer Worm.* IEEE Security & Privacy, Vol 1 No. 4, 2003.

[37] Levy, E., Arce, I.*The Spread of the Witty Worm.* IEEE Security & Privacy, Vol 2. No 4. 2004.

[38] Cowie, J., Ogielski, A., Premore, B., Smith, E., and Underwood, T. *Impact of 2003 blackouts on Internet Communications.*, Tech Report, Renesys, Nov, 2003.

[39] Markou, M., Singh, S. *Novelty detection: A review - parts 1 and 2.* Signal Processing, volume 83. 2003.

[40] Marais, E., Marwala, T.*Predicting the Presence of Internet Worms using Novelty Detection.* Technical Report, Computing Research Repository. 2007.

[41] Li, J., Dou, D., Wu, Z., Kim, S., Agarwal, V.*An internet Routing Forensics Framework for Discovering Rules of Abnormal BGP events.* ACM Special Interest Group on Data Communications (SIGCOMM) Computer Communication Review. Vol 35, Number 5, pg 57-66. 2005.

[42] Zhang, J., Rexford, J., Feigenbaum, J. *Learning-Based Anomaly Detection in BGP Updates.* Proceeding of the 2005 ACM Special Interest Group on Data Communications Workshop on Mining Network Data. 219 - 220. 2005.

[43] The Cooperative Association for Internet Data Analysis. *http://www.caida.org/*

[44] Menon, V., Pottenger, W.M. *A Higher Order Collective Classifier for Detecting and Classifying Network Events.* IEEE International Conference on Intelligence and Security Informatics, 2009. (Under review)

[45] Wang, K., Salvatore J. Stolfo.*One Class Training for Masquerade Detection.* International Conference on Data Mining (ICDM) Workshop on Data Mining for Computer Security (DMSEC) Melbourne, FL. 2003.

[46] M. Schonlau, W. DuMouchel. *Computer intrusion: Detecting masqueraders.* Statistical Science, Volume 16(1), pg:5874. 2001.

[47] Seo, J., Cha, S.*Masquerade Detection based on SVM and Sequence-based User Commands Profile.* ASIAN ACM Symposium on Information, Computer and Communications Security. 2007.

[48] Coull, S. E., Branch, J. W., Szymanski, B. K. Breimer, E. A.*Sequence Alignment for Masquerade Detection.* Technical Report. Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY. 2007.

[49] Smaldone, S., Bohra, A. Iftode, L.*Filewall: A Firewall for Network File Systems.* Proceedings of the 3rd IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC). 2007.

[50] Witten, I. H., and Frank, E.*Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco ISBN-13: 978-3446215337. 2005.

[51] Uno, T. *An Output Linear Time Algorithm for Enumerating Chordless Cycles*, 92nd SIGAL of Information Processing Society Japan, pg: 47-53. 2003.

[52] Rubin, F.*Enumerating All Simple Paths in a Graph*,IEEE Transactions on Circuits and Systems, Vol CAS-25 pg: 641-642. 1978.

[53] Valiant L.G. *The Complexity of enumeration and reliability problems*, SIAM Journal of Computing Volume 8(3), pg: 410-421. 1979.

[54] Roberts, B., Kroese, D.P. *Estimating the number of s-t paths in a graph.* Journal of Graph Algorithms and Applications Volume 11 (1), pg: 195-214. 2007.

[55] Bader, D. A., Madduri, K.*GTgraph: A Synthetic Graph Generator Suite.* For the 9th DIMACS Implementation Challenge. 2006.

[56] Ellard, D., Ledlie, J., Malkani, P., Seltzer, M. I. *Passive NFS Tracing of Email and Research Workloads.* USENIX Conference on File and Storage Technology (FAST). 2003