UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

ROLE BASED BEHAVIOR ANALYSIS

Ricardo Gonçalves Ramalho

MESTRADO EM SEGURANÇA INFORMÁTICA

Novembro 2009

UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

ROLE BASED BEHAVIOR ANALYSIS

Ricardo Gonçalves Ramalho

Orientador
Hyong S. Kim

Co-Orientador
Paulo Sousa

MESTRADO EM SEGURANÇA INFORMÁTICA

Novembro 2009

Resumo

Nos nossos dias, o sucesso de uma empresa depende da sua agilidade e capacidade de se adaptar a condições que se alteram rapidamente. Dois requisitos para esse sucesso são trabalhadores proactivos e uma infra-estrutura ágil de Tecnologias de Informação/Sistemas de Informação (TI/SI) que os consiga suportar. No entanto, isto nem sempre sucede. Os requisitos dos utilizadores ao nível da rede podem não ser completamente conhecidos, o que causa atrasos nas mudanças de local e reorganizações. Além disso, se não houver um conhecimento preciso dos requisitos, a infra-estrutura de TI/SI poderá ser utilizada de forma ineficiente, com excessos em algumas áreas e deficiências noutras. Finalmente, incentivar a proactividade não implica acesso completo e sem restrições, uma vez que pode deixar os sistemas vulneráveis a ameaças externas e internas. O objectivo do trabalho descrito nesta tese é desenvolver um sistema que consiga caracterizar o comportamento dos utilizadores do ponto de vista da rede.

Propomos uma arquitectura de sistema modular para extrair informação de fluxos de rede etiquetados. O processo é iniciado com a criação de perfis de utilizador a partir da sua informação de fluxos de rede. Depois, perfis com características semelhantes são agrupados automaticamente, originando perfis de grupo. Finalmente, os perfis individuais são comprados com os perfis de grupo, e os que diferem significativamente são marcados como anomalias para análise detalhada posterior. Considerando esta arquitectura, propomos um modelo para descrever o comportamento de rede dos utilizadores e dos grupos. Propomos ainda métodos de visualização que permitem inspeccionar rapidamente toda a informação contida no modelo.

O sistema e modelo foram avaliados utilizando um conjunto de dados reais obtidos de um operador de telecomunicações. Os resultados confirmam que os grupos projectam com precisão comportamento semelhante. Além disso, as anomalias foram as esperadas, considerando a população subjacente. Com a informação que este sistema consegue extrair dos dados em bruto, as necessidades de rede dos utilizadores podem sem supridas mais eficazmente, os utilizadores suspeitos são assinalados para posterior análise, conferindo uma vantagem competitiva a qualquer empresa que use este sistema.

Palavras-chave: análise de comportamento de rede, análise do perfil de utilizadores, comportamento de utilizador, fluxos de rede, fluxos aplicacionais, prospecção de dados, detecção de anomalias.

Abstract

In our days, the success of a corporation hinges on its agility and ability to adapt to fast changing conditions. Proactive workers and an agile IT/IS infrastructure that can support them is a requirement for this success. Unfortunately, this is not always the case. The user's network requirements may not be fully understood, which slows down relocation and reorganization. Also, if there is no grasp on the real requirements, the IT/IS infrastructure may not be efficiently used, with waste in some areas and deficiencies in others. Finally, enabling proactivity does not mean full unrestricted access, since this may leave the systems vulnerable to outsider and insider threats. The purpose of the work described on this thesis is to develop a system that can characterize user network behavior.

We propose a modular system architecture to extract information from tagged network flows. The system process begins by creating user profiles from their network flows' information. Then, similar profiles are automatically grouped into clusters, creating role profiles. Finally, the individual profiles are compared against the roles, and the ones that differ significantly are flagged as anomalies for further inspection. Considering this architecture, we propose a model to describe user and role network behavior. We also propose visualization methods to quickly inspect all the information contained in the model.

The system and model were evaluated using a real dataset from a large telecommunications operator. The results confirm that the roles accurately map similar behavior. The anomaly results were also expected, considering the underlying population. With the knowledge that the system can extract from the raw data, the users network needs can be better fulfilled, the anomalous users flagged for inspection, giving an edge in agility for any company that uses it.

Keywords: network behavior analysis, user profiling, user behavior, network flow, application flow, data mining, anomaly detection.

Acknowledgments

I would like to thank Professor Hyong Kim for his guidance and advice. His experience was vital to guide this project to a successful and timely conclusion. I would also like to thank Professor Paulo Sousa for his dedication in helping making this thesis an enjoyable reading experience.

To my colleagues from MSIT-IS class of 2008/2009, for all the companionship, debates and support throughout the course. To the colleagues from last year's course for their helpful tips and advice.

To John Payne for all the brainstorming sessions, lengthy discussions and advice. A special thanks to the group that travelled far and wide with me to make their thesis a unique experience.

To José Alegria for his support, incentive and trust in my abilities. To Tiago Carvalho and Luís Costa, for making the corporate – academic interface possible.

To my family and friends, who still remember me after these long 16 months!

Most of all, a very special thanks to my dearest Cláudia, for all her support, patience and dedication. This is just the remainder…

Contents

1 Introduction 1

1.1. Motivation 1

1.2. Contributions 2

1.3. Document structure 2

2 Background 5

2.1. Corporate environment 5

2.2. Dataset 6

2.2.1. Dataset characteristics 6

2.2.2. Limitations 8

2.2.3. Pre-processing 8

2.2.4. Classes 8

2.3. Data mining 9

2.3.1. Data 9

2.3.2. Clustering 12

2.3.3. Predictive modeling (classification) 15

2.3.4. Anomaly detection 15

2.3.5. Association analysis 17

2.4. Related work 17

2.4.1. Traffic classification 17

2.4.2. Data mining 20

2.4.3. User behavior 21

3 Model and architecture 23

3.1. Architecture 23

3.2. Profiler 23

3.3. User profile model 24

3.4. Roler 27

3.5. Roles model 28

3.6. Anomaly detector 31

3.7. Temporal analysis 31

3.8. Previous models 31

3.8.1. PCA / SVD 32

3.8.2. Overly simplistic model 32
3.8.3. Fixed categories on individual attributes (traffic/flows/duration) 33
3.8.4. Variable categories on all attributes 33
4 Results 35
4.1. Single analysis 35
4.1.1. Role visualization 39
4.1.2. Anomaly detector 42
5 Deployment 45
5.1. System components 45
5.1.1. Probes 45
5.1.2. Event repository (DB) 45
5.1.3. Profiler 46
5.1.4. Roler 46
5.1.5. Anomaly detector 46
5.1.6. Temporal analysis 46
5.2. Scalability analysis 46
5.2.1. User / distinct application ratio 46
5.2.2. Sparseness 47
5.2.3. Sparseness refinement 47
5.2.4. Projection 48
5.3. Privacy 48
6 Future work 49
7 Conclusion 51
8 References 53

List of Figures

Figure 1 – Pulso complex event processing architecture 5
Figure 2 – Total daily number of users 6
Figure 3 – Total daily traffic in dataset 7
Figure 4 – Total daily number of flows in dataset 7
Figure 5 – Process of knowledge discovery in databases 9
Figure 6 – PCA example 11
Figure 7 – Scree plot 12
Figure 8 – Clustering example 13
Figure 9 – Architecture block diagram 23
Figure 10 – Example pdf function that uses the normalized values presented in Table 4 25
Figure 11 – Tridimensional (duration, time, traffic) plot of dataset 32
Figure 12 – Scatterplot for application 520 33
Figure 13 – PDFs for user 2, application 58 35
Figure 14 – PDFs for user 252, application 58 36
Figure 15 – User/application spectrogram (high contrast) 36
Figure 16 – User/application spectrogram (high contrast) after clustering 37
Figure 17 – User/application spectrogram (high contast) detail after clustering 38
Figure 18 – Application 476 in cluster 3 39
Figure 19 – Application energy plot for all clusters 39
Figure 20 – Application energy plot for clusters 3 and 4 40
Figure 21 – Business application 476 in cluster 3 41
Figure 22 – Business application 476 in cluster 4 41
Figure 23 – Anomaly score plot 42
Figure 24 – User to distinct application ratio 47

List of Tables

Table 1 – K-means algorithm 13
Table 2 – DBScan algorithm 14
Table 3 – Market basket transactions example 17
Table 4 – Calculating a pdf function 25
Table 5 – A pdf function that exhibits compression 26
Table 6 – A pdf function calculated using a percentile 26
Table 7 – Dimensional unfolding example (k>=2, r=4, three attributes) 27
Table 8 – Role user distribution per class 38
Table 9 – Cluster 3 vs cluster 4 applications breakdown (over 50%) 40
Table 10 – Anomaly detection results 43
Table 11 – User / application sparseness (application granularity) 47
Table 12 – User / application sparseness (pdf slice granularity) 48

Abbreviations

BSP	Binary Space Partitioning
CPU	Central Processing Unit
dpi	dots per inch
I/O	Input/Output
IS	Information Systems
IT	Information Technologies
OS	Operating System
PCA	Principal Component Analysis
PDF	(Discretized) Probability Distribution Function
SVD	Singular Vector Decomposition

1 Introduction

1.1. Motivation

Corporations have always been under pressure to perform better than their competitors. Nowadays, that is truer than ever, as the world seems to evolve at an increasingly faster pace. Moreover, in a battlefield with conditions changing rapidly, corporations have to adapt quickly and be as efficient as possible. IT/IS systems have greatly helped on that field, empowering the workers to perform complex operations faster than before. However, how are those mechanisms being used?

Because of the highly dynamic world, agility and proactivity are two highly sought characteristics in workers. This means they actively search for new solutions and methods to solve problems they never encountered before. The internet, for example, can be seen as a huge knowledge repository that can be used to expedite problem solving and research. Also available on the internet are a myriad of tools (software) for almost every purpose conceivable. Proactive workers may download and use new tools that the corporation has not made available to their users. Of course, allowing that kind of access is a tradeoff with security. Nevertheless, even if they do not procure new tools, they can use existing (allowed) software in novel ways. For example, a server may be slow to respond during traditional work hours, making statistics collection from a supervisor a very inefficient endeavor. Some may chose to perform that task during off-hours so that they do not waste time. The application is the same, but it is being used in a different (temporal) way by different groups of users.

Consider this real example: in a client-server application, the users should fill a registration form with customer data. For existing customers, the lookup was supposed to be performed in another window, and when the correct data was located, copied into the registration form. However, the developers designed a registration form so dynamic, that it could be used directly to perform the query. Some users discovered this ‘accidental’ feature and began using it, as it was more practical. Moreover, as new users arrived, they were instructed by the older ones to do it this way, as it was the most efficient one. However, the form was intended for updates only, and not for queries. Albeit unnoticeable for the end-users, it caused locking problems on the underlying database, degrading performance for the rest of the users. On the other hand, some other users were unaware of this ‘feature’ and were using the application in the ‘intended’ way. In this example, different groups of users were using the same application in different ways.

The underlying question in these examples is: “What is the user’s behavior while using the IT/IS infrastructure?” The answer to this seemingly simple question may allow answering to several other, related issues. For example, what are their real needs? Do they use applications other than the ones the IT/IS support deemed necessary? How do they use the applications? In essence, in a very dynamic environment, it is difficult to correctly estimate the real user network resource requirements. Traditional approaches like over-provisioning are inefficient and expensive. Using server-side estimates may be overly complex as there are too many applications. In addition, this approach does not solve the problem of applications that use external servers (on other corporations’ extranets, or even in the internet).

A second aspect of the problem concerns corporate (re)organization and relocation. To remain competitive it is common for large corporations to modify its internal structure every couple of years, physically relocating departments and reassigning space. However, how to determine the network needs of the users that are being relocated? What do they do, how and when do they do it?

The third aspect involves trust. Even if Saltzer's principle of least privilege [1] could be completely enforced, some kind of privilege is still required by the users to execute their functions. What guarantee does the corporation have about that trust? In other words, how vulnerable is it to insider threats? And more importantly, how to detect it? They are very difficult to detect, and some approaches are based on auditing. A different approach is to track the flow of information to detect impending leaks [2]. Whatever the approach, it is noteworthy to mention that anomalous behavior may be indicative of insider threat, and should be investigated.

The fourth and final aspect concerns user compliance with the security policy with regard to applications. What applications are being used? Maybe they have found a new useful application. However, have they reviewed the licensing requirements? Alternatively, are they, even if involuntarily, executing malware on their machines?

The knowledge of user behavior can deal with these four fundamental aspects. We propose a model and architecture to characterize user network behavior and group them into similar roles. This characterization can be used to identify each role (and user) network requirements, and to identify deviant behavior.

1.2. Contributions

The main contribution of this work is a model and system architecture to characterize network user behavior and group it into similar roles. In addition, a prototype of the said model and architecture was implemented and allowed better evaluation and drawing conclusions instead of a purely theoretical work. Also as important, a real dataset was measured from a large IT corporation and used for the system evaluation. The fact that many works use synthetic sets only emphasizes the importance of using real data. In fact, it is our opinion that, since this work deals with human behavior, it would not be possible to correctly and effectively evaluate the model with simulated data. Finally, several visualization models were devised to display some of the multidimensional data structures.

1.3. Document structure

This work is organized as follows. Chapter 2 will detail the relevant background information. The corporate environment section contains information about the company and its systems that are related with this thesis. The dataset section analyses the real dataset that was used to implement and evaluate this work. Relevant concepts from the areas of statistics and data mining are also in this chapter. This chapter ends with related work on relevant areas.

Chapter 3 describes the system model and architecture. Each component is explained, as well as model properties. This chapter also contains the models of previous attempts at solving the problem.

In Chapter 4 the results are analyzed. The prototype was run against the real dataset, and the kind of information that the model contains is demonstrated. Anomaly detection was evaluated considering the underlying population.

Chapter 5 analyses deployment into a production scenario. Every component is analyzed regarding the resource requirements and bottlenecks. The scalability of the entire system, regarding a large-scale deployment is also analyzed. Finally, due to the nature of the data contained in the model, privacy issues are addressed. A deployment scenario that minimizes privacy exposure is presented.

This work concludes with future work and evolution on Chapter 6 and the conclusions on Chapter 7.

2 Background

This chapter details the relevant background information. The corporate environment section contains information about the company and its systems that are related with this thesis. The dataset section analyses the real dataset that was used to implement and evaluate this work. We rely on concepts from the areas of data mining and statistics, and a primer on both can be found on the corresponding section. This is, by no means, a complete analysis of data mining techniques. It is intended as a reference section, so that the relevant concepts are not introduced on first use, breaking the flow of this document. This chapter ends with the related work on relevant areas.

2.1. Corporate environment

The background of this project is a large IT Company that has facilities and workers spread throughout the country, even if more concentrated in urban areas. To help monitor and manage its IT infrastructure, it deployed a Complex Event Processing [3] system named Pulso [4], that performs data acquisition, processing, transformation and display. Its architecture is depicted in Figure 1.

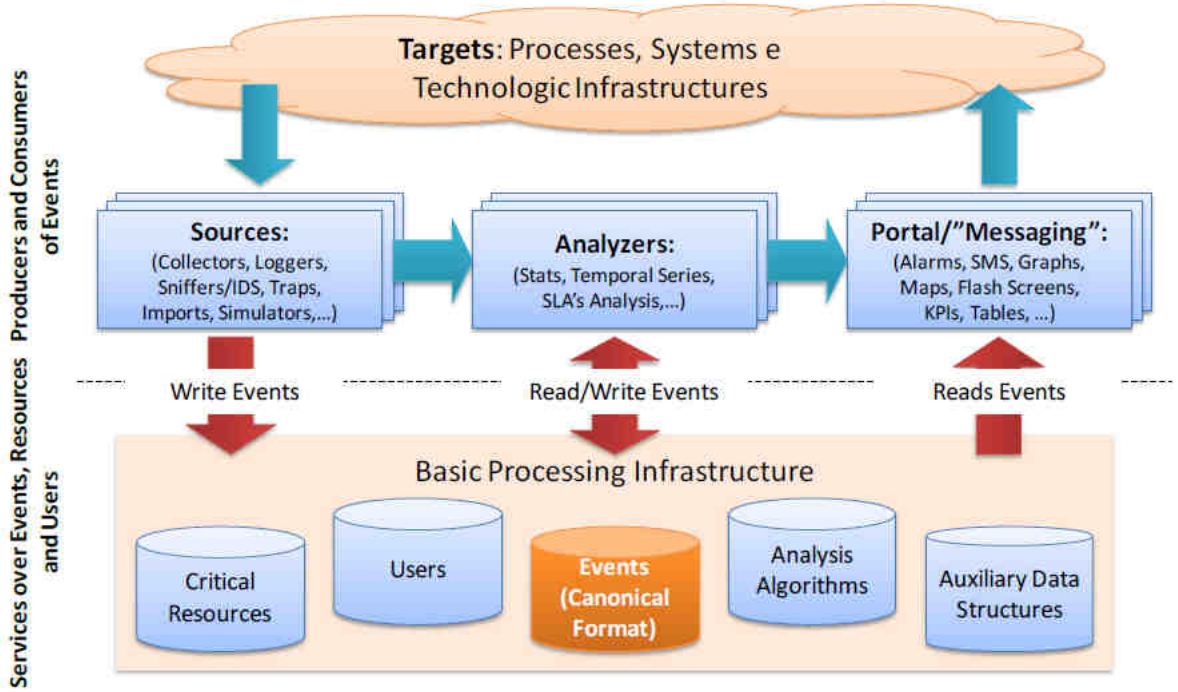

Figure 1 – Pulso complex event processing architecture

The Pulso system is composed of three kinds of entities. Data sources produce events that are sent and stored in the event repository, in a canonical format. Examples of data sources are active network probes, sniffers, application-generated events, etc. The analyzers process the raw events into higher-level events, extracting relevant information from data. Finally, the display entities

generate or display the available information into human-readable formats, like alarms, maps, graphs, tables, performance indicators, visual displays, web portals, etc.

The latest evolution of Pulso features workstation probes. These obtain flow information tagged with the source application and the account under which the application is executing. This kind of detail is rarely available, and it was the main drive for this project – to infer and group user behavior based on their tagged network flows.

2.2. Dataset

For this project, a real dataset with two months worth of recent data, from April to June 2009, was made available. The dataset was obtained on a call center and contains tagged network flow information. A flow is defined in the usual way, the packets involved in a connection between a source and destination, usually characterized by the 4-tuple <source IP, destination IP, destination port, transport level protocol (TCP/UDP)>. The flows are tagged, and the tags contain the application that originated (or received) the flow, as well as the account under which it was launched. The information available for each flow is the start timestamp, the duration and total bytes transferred (per direction, and per transport protocol). Therefore, the information available are flow records of the kind <source IP, destination IP, destination port, transport level protocol (TCP/UDP), **application, user account**, start timestamp, duration, bytes transferred>. The information in bold is the one that is unique on this work, only available because of the workstation probes.

2.2.1. Dataset characteristics

The transformation from accounts to users is rather simple. Domain accounts translate directly into users. The remaining local accounts are related to system processes, and are designated System accounts.

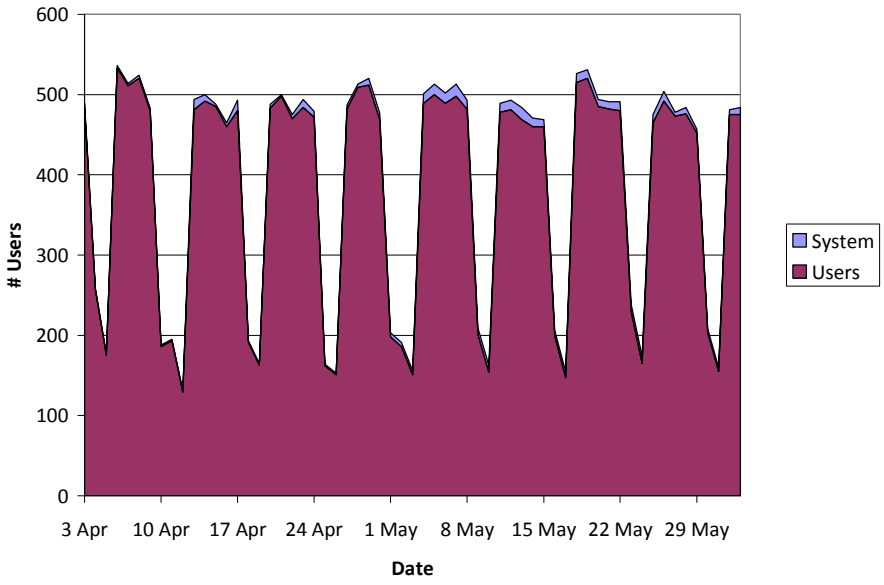

Figure 2 – Total daily number of users

As it can be seen on Figure 2, the dataset has, on average, 160 distinct users per day on weekends and 490 on work days. A small percentage is system users. Figure 3 shows that total traffic averages from 3GB/day on weekends to 7GB/day on work days. System processes generate a little additional traffic.

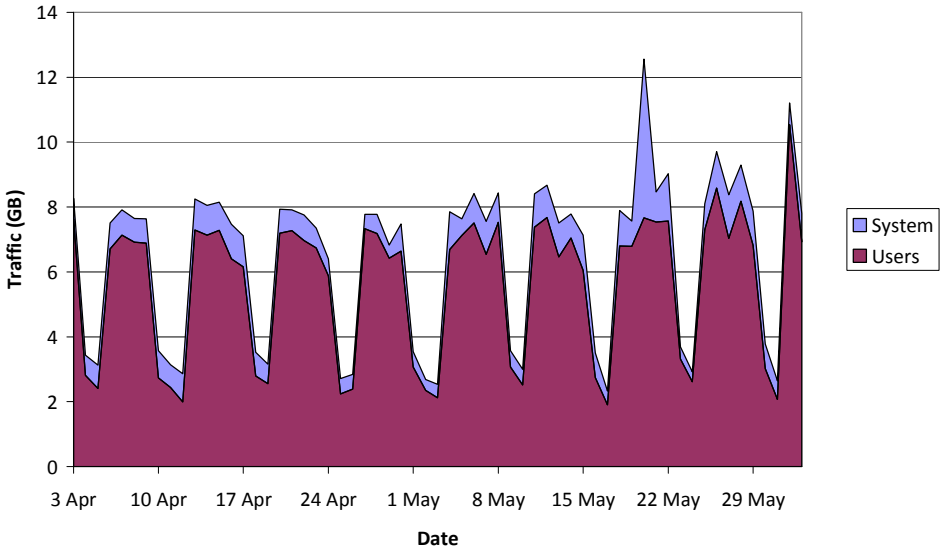

Figure 3 – Total daily traffic in dataset

In Figure 4, the number of distinct user flows is, on average, 55K/day on weekends and 130K/day on workdays. System processes increase these figures by 40%.

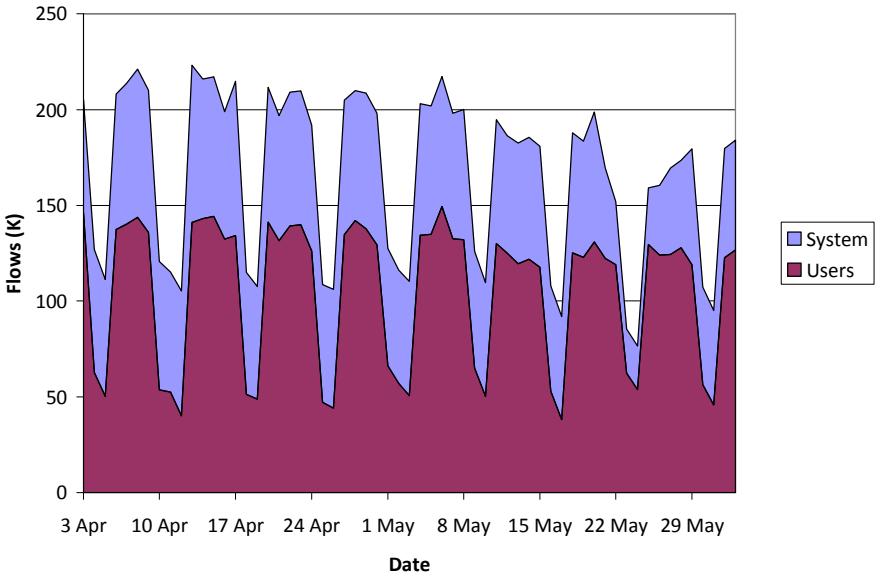

Figure 4 – Total daily number of flows in dataset

Ignoring all the system processes flows, the entire two month dataset has 1131 distinct users and 297 distinct applications (at the binary level). From these statistics, we can conclude that the dataset has enough size and information to be statistically significant.

2.2.2. Limitations

This dataset is all the information available to execute and evaluate this project, no additional information can be obtained. In addition, the probe system is already in place and cannot be modified. However, modifications and suggestions that result from this work can influence the next versions.

2.2.3. Pre-processing

'Users' are domain accounts that univocally correspond to human workers (there are no shared accounts). 'System' accounts refer to local workstation accounts related to OS tasks. By analyzing the data, system accounts were filtered from the dataset for two reasons. First, the applications that are directly run by users are more likely to mirror their behavior. After all, these are the ones they are interacting with. Second, system accounts will most likely be performing OS-relevant functions, and would generate noise for our analysis. Third, only a few tasks performed under system accounts might indirectly mirror human behavior, by being related to user applications. These are a minority, and this benefit is probably insignificant when compared to the noise they could add. For these reasons, they were filtered.

2.2.4. Classes

The user population was identified into three distinct 'classes' or categories. Note that in a single class there can be users from different departments. The three classes are:

- Class 0 – Callcenter Operators. These users have no special privileges, and should be performing a well-defined subset of tasks;
- Class 1 – Callcenter Supervisors. Supervisors monitor callcenter sections, and therefore have greater responsibility and privileges. Their tasks are less clear, since they have to solve all the problems that may arise;
- Class 2 – Others. This category includes users from tech support and specialized teams that may have administrative privileges and almost unlimited flexibility in the tasks they perform.

The distribution from these classes is far from even; 90% are operators, 5% are supervisors and just 3% are others.

- Class 0 (Operators): 1035/1131 (91,5%)
- Class 1 (Supervisors): 60/1131 (5,3%)
- Class 2 (Other): 36/1131 (3,2%)

2.3. Data mining

Data mining can be described as the process of extracting information from data. The main characteristic that tells it apart from traditional information retrieval techniques is the enormous sizes of the datasets. Traditional techniques may be unsuitable due, not only to this, but also because of the high dimensionality, heterogeneity and distributed nature of the data. To solve these problems, data mining is a confluence of areas, drawing ideas from machine learning, artificial intelligence, pattern recognition, statistics, and database systems. It is also related with Knowledge Discovery in Databases (KDD), the overall process of converting raw data into useful information. The steps of the KDD process, according to [5], are depicted on Figure 5. There are three main stages. The data preprocessing stage consists in reducing the available data to a manageable and meaningful set, i.e., selecting the relevant features, normalizing the data, etc. The data mining stage consists of extracting relevant information from this data set. The final stage may include manual tasks, and involves representing the relevant information into human-understandable form.

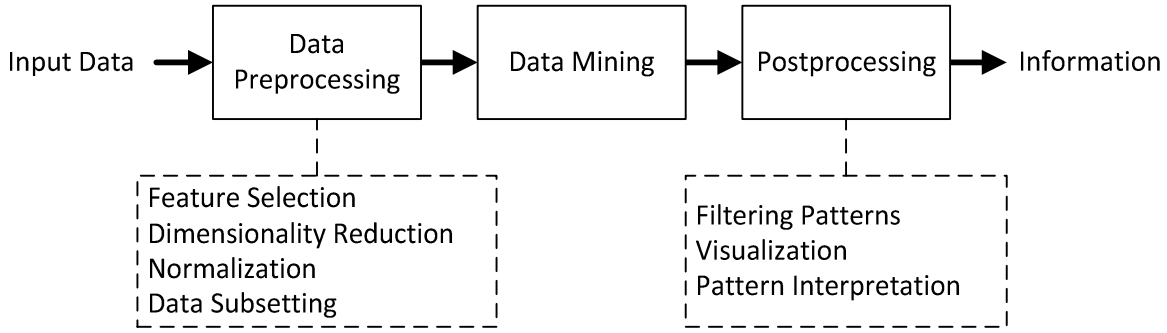

Figure 5 – Process of knowledge discovery in databases

The core data mining tasks are Clustering, Association Analysis, Predictive Modeling (Classification) and Anomaly Detection. These fall into two broad categories:

- Predictive tasks: Predict the value of some attributes (target variable) based on the value of other attributes (explanatory variables). Tasks in this category are Predictive Modeling (Classification) and Anomaly Detection;
- Descriptive Tasks: Derive human-interpretable patterns that summarize the underlying relationships in data. Clustering and Association Analysis are usually classified as descriptive tasks.

In the next section, we will first analyze the properties of data itself. Then, we will briefly discuss these tasks, with greater emphasis on the most relevant for our work.

2.3.1. Data

Evidently, data mining is about data. Data can be seen as a collection of data objects, each with a collection of attributes that characterizes it. The values the attributes can have depend on their type. Four properties are used to describe attributes, related with the operations than can be performed with them:

- Distinctness: $=$ and $\neq$. With only these two operations, attributes can be either equal or different from one another.
- Order: $<$, $\leq$, $>$ and $\geq$. These operations allow the values of an attribute to be ordered.
- Addition: + and –. These operations allow the values of an attribute to be added and subtracted.
- Multiplication: * and /. These operations allow the values of an attribute to be multiplied and divided.

With these properties, attributes are classified into four types. Each type contains all the properties of its predecessor:

- Nominal (distinctness). These attributes can only be compared against each other. Examples are ZIP codes, eye color, gender, etc.
- Ordinal (order and distinctness). These attributes also have a natural order, like happens with grades, street numbers, scales, the Mohs scale of mineral hardness, etc.
- Interval (addition, order and distinctness). These attributes can be added (and subtracted), which happens, for example, with calendar dates, temperature in Celsius or Fahrenheit.
- Ratio (multiplication, addition, order and distinctness). These attributes have an absolute zero value, and therefore can be multiplied (and divided), like what happens with age, mass, length, temperature in Kelvin, etc. Therefore, it makes sense to say, for example, that one person weighs twice as much as another person, but it makes no sense to say that a temperature of 20 Celsius is twice as warm as a temperature of 10 Celsius (because the Celsius scale has no absolute zero). However, on the Kelvin scale, 200 degrees can indeed be said to be twice as warm as 100 degrees.

It is also common to group nominal and ordinal attributes as categorical (or qualitative) attributes. Similarly, interval and ratio attributes can be labeled as numeric (or quantitative) attributes.

Nevertheless, the attributes can also be classified by a different property, the number of values they can have:

- *Discrete attributes* have a finite or countable infinite set of values. These attributes can be categorical, such as the examples before. Binary attributes are a special case of discrete attributes with just two values.
- *Continuous attributes* are those whose numbers are real attributes.

Both of these classifications are important because most techniques can only handle certain type of attributes, and it is common to refer to them by any of these classifications.

Data sets

All data sets have three characteristics that have a significant impact on the data mining techniques that can be used: dimensionality, sparsity and resolution.

The *dimensionality* of a data set is the number of attributes that the objects in that data set possess. The difficulty in analyzing high-dimensional data is referred as 'the curse of dimensionality'. To solve this, dimensionality reduction techniques are used, and some will be discussed later.

Sparsity relates to the ratio of the dataset object attributes that have a non-zero value over the entire data set object attributes. If this ratio is low, the dataset is labeled sparse. In practical terms, this is an advantage since usually only the non-zero attributes have to be stored and manipulated.

Resolution influences the size of the dataset, and, consequently, which properties will be discernible, i.e., the patterns on data depend on the level of resolution. If the resolution is too high, the pattern may be hidden in noise, if it is too low, the pattern may disappear altogether. For example, a document scanned at 100 dpi is smaller, and may allow individual characters to be recognized. The same document scanned at 100 times that resolution, 10K dpi, will be much larger. The type of printing technology and paper texture can be discerned, but the fact that it contains text may not be detected.

Dimensionality reduction techniques

Principal Component Analysis (PCA) [6] and Singular Vector Decomposition (SVD) [7] are two widely used techniques for dimensionality reduction. PCA differs from SVD in the fact the PCA removes the mean of the data and SVD does not.

Both involve a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The first principal component (dimension) tries to capture as much of that variability as possible. The second dimension will be orthogonal to the first, captures as much of the remaining variability, and so on. The resulting set of dimensions constitute a new multidimensional space where the original data is projected. The idea is to capture as much of the variability of the data as required in a lesser dimensional space. The original dataset can then be projected into this reduced-dimensionality space. Since it has fewer dimensions, it is likely that more data mining algorithms can be applied to the data set. Note that in the end of processing, the data can be projected back into the original space, where it has meaning.

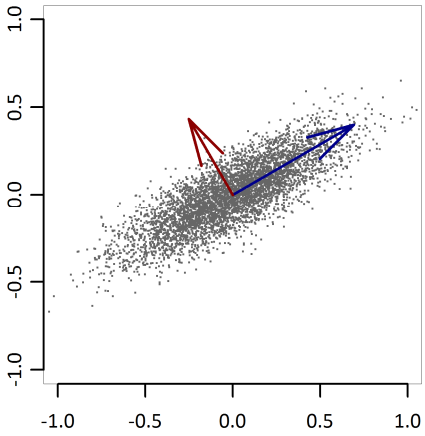

Figure 6 – PCA example

Take Figure 6 as an example. The two vectors clearly point to the directions of greater variability. If that space had to be reduced to just one dimension, the longest vector could be used, and a projection on that (uni-dimensional) space would account for most of the variability.

Another common visualization method related with dimensionality reduction is the scree plot, shown in Figure 7. A scree plot shows the relationship between the principal components and the variance explained by each of them. They are sorted by explained variance (Y-axis), and the X-axis contains the principal components. A scree plot helps the analyst visualize the relative importance of the principal components (also called factors) — a sharp drop in the plot indicates that subsequent factors are ignorable, because they explain less variability. In Figure 7, the most important components are the first two, because they are clearly higher than any of the others. It is also common to select the principal components up to the 'knee point' of the curve.

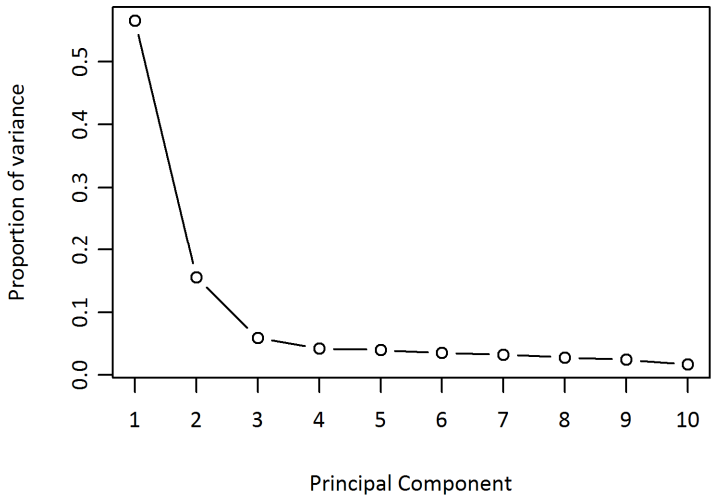

Figure 7 – Scree plot

In short, PCA has some interesting characteristics:

- Tends to identify the strongest patterns in the data. Can be used as pattern finding technique;
- Often, most of the data variability can be captured by a small fraction of total dimensions. After this dimensionality reduction, low-dimensionality techniques can be used;
- If the noise is weaker than the patterns, dimensionality reduction can eliminate much of the noise.

2.3.2. Clustering

The goal of cluster analysis is to divide data into groups (clusters) that are meaningful and/or useful. For that, the clustering algorithm needs a similarity measure, a function that calculates the degree to which two objects are alike. The Euclidean distance is a very simple and popular similarity distance for continuous attributes, but there are others, like the Mahalanobis distance, the cosine similarity or the Jaccard coefficient.

Any clustering algorithm will try to group data points such that intra-cluster similarity is maximized and inter-cluster similarity is maximized. In other words, create clusters such that data points in one cluster are more similar to one another and data points in separate clusters are less similar to one another.

Consider the example in Figure 8. The algorithm used the Euclidian distance as its similarity measure, so the idea was to group points that were close to one another. The data set was grouped into five groups, indicated by the different shapes (and colors). The goal was achieved, as five distinct groups are evident.

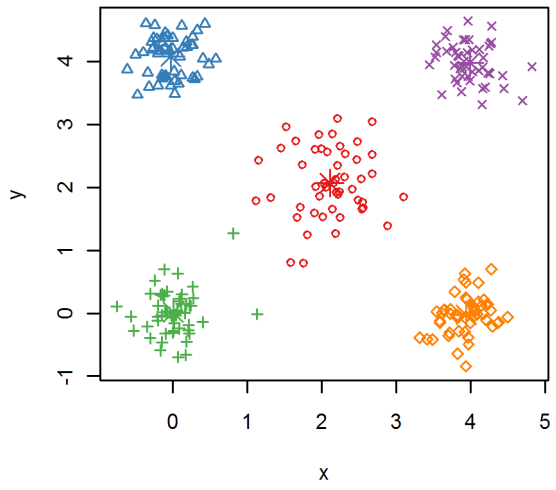

Figure 8 – Clustering example

Next, three clustering algorithms will be presented: K-means, X-means and DBScan.

K-means

K-means [8] is one of the oldest and most widely used clustering algorithm. It is a prototype-based clustering algorithm that can be applied to objects in a continuous n-dimensional space. In prototype-based clustering algorithms, a cluster is a set of objects that is closer (more similar) to the prototype of that cluster, than to the prototype of any other cluster. In K-means, the prototype is the centroid, i.e., the mean of a group of points. The algorithm, quite simple, is shown on Table 1.

1. Select K points as the initial centroids
2. Repeat
3. Create K clusters by adding the points to its closest centroids
4. Recalculate the centroids for each of the K clusters
5. Until the centroids do not change

Table 1 – K-means algorithm

Despite its simplicity and effectiveness, it is not without issues. First, K (the number of clusters) has to be manually specified beforehand. In addition, the selection of the initial centroids affects the outcome of the algorithm. The algorithm may converge to a local minimum, and not to a global minimum. Several workarounds can be used, like randomly generating centroids, running that algorithm multiple times and choosing the best run (the one with less total error). A better approach is the one used in X-means, which we will analyze in the next chapter.

K-means does have modest time and space requirements. Let m be the number of points, n the number of features and t the number of iterations to converge. Its space complexity is O((m+K)n), because only data points and centroids are stored. Time complexity is O (t*K*m*n), but since t is usually small, and if K<<m, it will be linear with m. In short:

- K-means strengths: Simple, efficient, even with multiple runs;
- K-means weaknesses: Not suitable to all types of data: non-spherical clusters or clusters of different sizes and densities; has trouble clustering data with outliers, although outlier detection and removal can help.

X means

X-means [9] is a Carnegie Mellon evolution of the popular K-means algorithm that addresses two of its shortcomings: it improves computational scalability and determines an 'optimal' number of clusters K. This algorithm is much faster than repeatedly using K-means for different values of K, a common practice for automatically determining K. The computational scalability improvements are due to the data structure used (multiresolution kd-trees, a special case of BSP-trees) and blacklisting (considering only the needed centroids for each region).

The optimum K is determined by recursively splitting each centroid into two or more, and evaluating if the resulting subclusters are better than the single one – a technique known as bisecting K-means. The algorithm uses the Bayesian Information Criterion (BIC) scoring function as its optimal measure. Nevertheless, it still requires a range of K ([kmin, kmax]) as input to determine the optimum K.

DBScan

DBScan [10] is a density-based clustering algorithm. These algorithms locate regions of high density that are separated from one another by regions of low density. The density of a point is the number of points inside a radius Eps (a configurable parameter). The results of this type of algorithm depend heavily on Eps. Consider the extreme cases: If Eps is too large, the density will equal m, i.e., all points are within Eps. On the other hand, if it's too small, the density will equal 1 (only that point is within Eps).

The algorithm, shown in Table 2, uses three types of points:

- Core Points are the ones in the interior of a cluster. A point is considered a core point if the number of points inside Eps exceeds a threshold MinPts (the other configurable parameter).
- A Border Point is a point that is not a core point, but is in the neighborhood of at least one border point
- Noise points are all the others, i.e., points that are neither core nor border.

1. Label all points as core, border or noise 2. Eliminate noise points 3. Create an edge between all core points within Eps of each other 4. Make each group of connected points a separate cluster 5. Assign each border point to one of the clusters of its core points

Table 2 – DBScan algorithm

Let m be the number of points. Time complexity of DBScan is O(m*time to find points in the Eps neighborhood). The worst case is $O(m^2)$, but using kd-trees and in low-dimensional spaces it can be O(mlog(m)). Space complexity is just O(m), since only small amount of data is kept for each point: cluster label and type (core, border, noise). Summing up:

- DBScan Strengths: Classifies points as outliers, not all have to forcibly belong to a cluster; for the same reason, it's resistant to noise; handles clusters of arbitrary shapes and sizes;
- DBScan Weaknesses: Does not handle well clusters with widely varying densities; it also does not handle well high-dimensional data, since density is more difficult to define; expensive when computing nearest neighbor requires computing all pair-wise proximities, as it is high-dimensional data.

2.3.3. Predictive modeling (classification)

Classifiers require a collection of records (designated a training set), where each record contains a set of attributes, and one of the attributes is the class. The purpose of a classifier is to classify previously unseen records (assign a class) as accurately as possible. That is accomplished by finding a model for the class attribute as a function of the values of other attributes, i.e., class = f($attribute_1$, $attribute_2$, …, $attribute_n$). To determine the accuracy of the model, test sets (previously unseen records) are used. Usually, a given data set is divided into training and test sets, with the training set used to build the model and test set used to validate it.

Classification techniques are most suited for datasets with binary or nominal categories and less effective for ordinal categories. There are many techniques for building classifiers, each resulting in a different type of model: decision trees, rule-based, nearest-neighbor, Bayesian, artificial neural networks, support vector machines, ensemble methods, etc.

Classifiers are evaluated based on metrics derived from its classification accuracy. This means that some of the models are not designed to be easily interpreted by humans, and use the black box approach, in the sense that no information can be extracted from them. Neural networks are an extreme example of that.

2.3.4. Anomaly detection

The idea behind anomaly detection is simple: find objects that are different from most other objects. A variety of approaches exist, all trying to capture the idea that an anomalous object is unusual or in some way inconsistent with the other objects. The three main approaches for anomaly detection are:

- Model-Based – Build a model, anomalies are the ones that don't fit well into the model;
- Proximity-Based – Anomalous objects are distant from most other objects;
- Density-Based – Objects in low density regions are relatively distant from their neighbors and can be considered anomalous.

Class labels are labels that indicate if a data point is anomalous or normal. Considering the class label requirements, each technique can be classified as supervised, unsupervised and semi-supervised.

Supervised anomaly detection requires a labeled training set with both anomalous and normal objects. In unsupervised anomaly detection, no class labels are available. On semi-supervised, only normal labels are available in the training set (no anomalous labels).

Chandola's recent and thorough survey on anomaly detection [11] is a new reference in this area, much more complete than the previous overviews, for example, Patcha [12]. Patcha's was more focused on Intrusion Detection, while Chandola's covers a broader spectrum of applications. Note that anomaly detection is not our primary goal, and, therefore, not all techniques are available to us.

Classification-based anomaly detection involves training a classifier with normal (or abnormal) data, so that it can recognize (classify) anomalies. However, most of the classifiers require supervised data in the training phase, i.e., the supplied data must be labeled (as 'normal' behavior, or as 'abnormal' behavior). Nearest-neighbor techniques assume that normal data instances occur in dense neighborhoods, while anomalies occur far from their closest neighbors. These techniques (distance to k^{th} nearest neighbor and relative density) are somewhat similar to the ideas behind clustering. Clustering-based anomaly detection is of particular interest, since we are grouping similar behavior into clusters. The available techniques rely on one of three basic assumptions:

A1: Normal data instances belong to a cluster, while anomalies do not. Under this assumption, any clustering algorithm that does not forcibly cluster all elements can be used, like DBSCAN. A disadvantage of these techniques is that they may not be optimized for finding anomalies, since their main aim is to find clusters.

A2: Normal data instances lie closer to their centroids than anomalies. Techniques that rely on this assumption first use any clustering algorithm, and then, the distance to its closest centroids is its anomaly score. K-means clustering, Expectaction-Maximization and Self-Organizing Maps (SOM) have been analyzed by Smith et al. in [13].

One weakness of this approach is that if the anomalies form clusters by themselves, these techniques will not detect them. In other words, anomalous clusters are not detected. To tackle this problem, a third category of clustering based techniques has been used, which relies on the following assumption:

A3: Normal data instances belong to large and dense clusters, while anomalies either belong to small or sparse clusters.

Although not considered by the authors, if the number of clusters is acceptable and human-understandable, manual classification can be a perfect detector of such anomaly clusters.

Statistical anomaly detection is also worth mentioning. The underlying assumption is that normal data instances occur in the high probability regions of a stochastic model, while the anomalies occur in the low probability regions. These techniques fit a statistical model to the data, and then apply a statistic inference test to determine the probability of an instance belonging to the model or not.

Parametric techniques assume that the normal data is generated by a parameterized parametric distribution. Non-parametric techniques do not require that the model is known a priori, it is defined from the data. These techniques typically make fewer assumptions regarding the data.

Histogram-based techniques construct histograms to maintain a profile for each attribute of the data. Kernel function based techniques use a function to estimate the actual probability density function. Parzen windows estimation [14] is a known and proven function. Note that kernel-based techniques can potentially have quadratic time complexity in terms of data size.

2.3.5. Association analysis

The goal of association analysis is, given a set of records each of which contain some number of items from a given collection, to produce dependency rules which will predict occurrence of an item based on occurrences of other items. This area first originated to deal with customer purchase data, designated market basket transactions.

Transaction ID	Items
1	{**Bread**, Milk}
2	{Milk, **Apples**, Juice, Cola}
3	{**Bread**, Milk, **Apples**, Juice}
4	{**Bread**, Milk, **Apples**, Cola}
5	{**Apples**, **Bread**, Eggs, Juice}

Table 3 – Market basket transactions example

For example, considering the transactions on Table 3, it might be concluded that {Bread} -> {Apples}, i.e., clients that buy bread are very likely to buy apples. This exemplifies the type of conclusions that can be drawn from association analysis. A very important aspect of these algorithms is that they only consider *sets* of categorical data, resulting in binary attributes. That is, the amounts of each item of a set are ignored. This has an impact on which domains these techniques can be applied.

2.4. Related work

Our work is a novel approach that applies existing and proven techniques towards a new goal, user network behavior analysis. Evidently, it is a multidisciplinary project that builds on several relevant fields, namely, traffic classification, data mining and user behavior. The related works are grouped by their major field, and analyzed in the next subsections.

2.4.1. Traffic classification

The goal of traffic classification is to identify and/or categorize network traffic, by analyzing some of its properties. The most common goal is to determine which application generated that traffic. While this is clearly not the main goal of this project, since the type of traffic and even the application is already known, this area of research is very relevant. The techniques and methods used to group and classify traffic are the foundation for grouping the users, since our working data consists of network data flows.

Botminer [15] is a botnet detector using network traffic. Although the goal is quite different from our own, the techniques used are very interesting and similar to the ones we developed. Under the assumption that bots within the same botnet are likely to behave similarly, the technique involves clustering hosts using features of their behavior. First, flows are grouped into <source ip, destination ip, destination port, protocol (tcp/udp)> groups, per day. The relevant attributes considered are

number of flows per hour, number of packets per flow, average bytes per packet, and average bytes per second. Afterwards, for each of these four random variables, an approximate version of their distribution is computed by binning – dividing the entire sample set into groups of interval ranges, called bins. However, instead of using fixed-size binning, n quantiles between 5% and 90% are computed, and their values used as the bins limit. The last bin will contain all samples greater than the last quantile (90%). In effect, this technique is providing variable-sized bins, maximizing the information retained by the approximation.

Clustering is performed in two steps. Since the previous binning technique generates many dimensions (number of bins * number of attributes), and the datasets also have high cardinality, a first, coarse-grained clustering is performed. For dimensionality reduction, only the mean and variance of the each distribution is used. Afterwards, for each cluster, a second clustering step is performed, this time with all the dimensions. The clustering algorithm used is X-means. Also noteworthy is the fact that two different data sources are used and clustered (network flows and activity). Afterwards, the clusters are correlated, and each host will receive a high score if it has performed multiple suspicious activities, and if other hosts in the same cluster exhibit the similar behavior.

In [16], Erman et al. used clustering algorithms to perform traffic classification using only traffic statistics at the transport layer level. Classification techniques using transport-layer statistics rely on the fact that different applications typically have distinct behavior patterns when communicating on a network. For example, a large FTP file transfer would have a larger connection duration and packet size than an instant-messaging client sending sporadic short messages to other clients. Clustering analysis is one of the methods for identifying classes among a group of objects. They compared K-means, DBSCAN and AutoClass algorithms on two empirical packet traces. For evaluation, the ratio of true positives over number of samples was used. Autoclass achieved the best accuracy, but only by a small margin, and it is almost two orders of magnitude slower than the other two. This factor also limits the number of samples it can handle in a reasonable amount of time. DBSCAN had the highest precision (ratio of true positives to false positives), and has the ability to label samples as noise.

BLINC [17] (BLINd Classification) uses a different approach to traffic classification. Instead of classifying individual flows, it associates hosts with applications and then classifies their flows accordingly. The argument is that observing the activity of a host provides more information about the nature of the applications of that host. Note that BLINC only classifies hosts according to 11 categories; it does not identify individual applications. Nevertheless, it does not use payload information or well-known port information. The behavior of each host is captured at three levels: social, network and application.

The social level captures the interaction with other hosts, and its popularity (the number of other host it communicates with). This is used to detect groups, or, communities of nodes, which are hosts that interact with each other. This grouping is performed using cross-association algorithms.

At the functional level, it is determined if a host is a provider or a consumer of a service, or if it participates in interesting communities. This is done by using the number of source ports a particular host uses for communication. A server is likely to use a single source port in most of its flows, while a client is more likely to use many (ephemeral) source ports.

The application level captures transport layer interactions and the final classification is refined. The host behavior is captured by matching all the levels against a library of empirically derived patterns, called graphlets. It is not mentioned how these patterns are derived, but it is highly likely a manual task, which is disappointing.

In conclusion, BLINC is a complex multilevel classification approach that has a reasonable level of accuracy and completeness.

Roughan et al. proposed a statistical signature-based approach [18] to solve the traffic classification problem. The purpose was to classify traffic into a reduced set of classes (interactive, bulk data, streaming and transactional). For each class, few applications (typically one) were chosen to train the model, using supervised learning, a predictive modeling technique. The methods considered were K-Nearest Neighbor and Linear Discriminant Analysis. The most relevant traffic features used in the model were average packet size and flow duration. Nevertheless, the authors had to include an additional feature, packet inter-arrival variability, to more accurately distinguish streaming traffic from data transfer traffic. Using 10-way cross-validation for evaluation, the results error rate was under 10% for the training applications, but higher for new applications. It seems that more work needs to be done.

In ACAS [19], Haffner et al. also tackled the traffic classification problem by creating signatures. However, their goal was to determine which application a flow belongs to by inspecting application layer information only (payload). Using supervised learning techniques (Naïve Bayes, AdaBoost and Maximum Entropy), each classifier was trained using labeled data. Results showed that the first 64 bytes of each flow were sufficient to classify the seven applications analyzed with a maximum error not exceeding 0.6%. Two of the applications were encrypted streams, namely https and ssh. In short, ACAS is using machine-learning techniques to distinguish the header of each flow.

What is interesting in ACAS is the technique used to convert the payload data. Since these classifiers are known to perform well on binary data, a discrete byte encoding technique was used to convert the payload data into binary features. The first n-bytes of a flow f are converted into a feature vector v with n*256 elements. The value of each byte sets the value of the corresponding vector position to one. All the others remain at zero. This encoding also has the important property that all byte values are equidistant (based on the Euclidean distance). That is, each byte value is either equal (distance=0) or different (distance=1) from each other. If the first n-bytes of f were used directly, for example, byte values 23 and 25 (distance=2) would be considered closer (more similar) than byte values 50 and 80 (distance=30). This would introduce an unwanted bias in the results.

The problem that Tan et al. address in [20] is different; the goal was to group hosts into similar groups, based on their connection patterns. The number of connections is not used, only the fact that a host has contacted another. The grouping problem was solved by reducing it to a bi-connected component graph problem. However, the interesting part was that the number of groups produced was deemed too large. A subsequent phase reduced the number of groups by merging groups that where similar, according to the similarity function and number of connections. This phase had tuning parameters that had to be set to produce the desired results. Also noteworthy, was the notion of identifying group evolution over time. That is, groups formed by subsequent runs of the algorithms may have different ids, while being functionally the same. This role correlation

algorithm allowed tracking group variations over time. This is an important issue and solution, as grouping algorithms not always generate groups with the same id.

2.4.2. Data mining

The work by Moore, Han et al. on [21] deals with many of the challenges that we also faced, namely, when the dimensionality of the feature space becomes high relative to the size of the document space. The domain was to categorize web pages and extract the most relevant features (in this case, words). One of the methods is ARHP [22], which uses association rules discovery and hypergraph partitioning. The other one is Principal Component Analysis (PCA) Partitioning Algorithm, which cuts the space of documents with a hyperplane passing though the arithmetic means of the document and normal to the principal direction. The process is repeated recursively, creating a tree like hierarchy along the principal vectors. The algorithms were compared using an entropy measure, and both methods performed better than the traditional clustering algorithms analyzed: AutoClass and Hierachical Agglomeration Clustering (HAC).

However, a more detailed analysis and their subsequent work on [23] clarifies that their model is suited towards frequent itemset mining, and not "clustering" in the traditional sense. More importantly, their scheme does not naturally handle continuous variables, which is the type of data we have on our work. They did develop a new algorithm called Min-Apriori [24] that operates directly on continuous variables without discretizing them. However, it is not clear if this variation was used in more recent works, like PACT [25].

Henderson et al. focused on analyzing user behavior in networked games [26], namely, Half-Life, Quake and Quake 3. Some results were expected, for instance, the number of players exhibit strong seasonal (time-of-day) variation. Player's session duration time fit an exponential distribution, and interarrival times fit a heavy-tailed distribution. In addition, the number of players in a session affects new players' decision to join or not a server, corroborating the empirical idea that players do not want to join empty servers. Also interesting were the techniques used for the analysis. The temporal autocorrelation function (ACF) and ARIMA (Autoregressive Integrated Moving Average) models [27] were used to examine these network externality effects.

Park and Giordano use anomaly detection to detect insider threats in [28]. They define the users' behavior using frequency patterns of their system actions (Search, Send, Copy). Then, as users have roles assigned to them, a pattern is inferred for each role. To detect anomalies, their technique involves two facets: role-level comparison, and individual comparison. Role-level comparison, matches each user's behavior against that of its role (spatially). Individual comparison matches each user's behavior against their own behavior while performing the same tasks in the past. Unfortunately, the implementation used a controlled simulation with synthetic data sets. Nevertheless, this dual-base comparison (against the group, and against the past) is quite interesting.

The work by Lakhina in [29] is relevant, because of the usage of PCA for network traffic flows analysis. This was an example of using PCA to solve the "curse of dimensionality". The analysis involved analyzing origin-destination flows over time in a network. In the datasets considered, there were fewer than two hundred dimensions. This is a high value, but not as high as the one we came across in our model. Nevertheless, just between 5 and 10 principal dimensions were enough to

accurately approximate the network flows. Another interesting conclusion was the classification of the eigenflows (network flows projected in some principal components) in three distinct categories: seasonal, spike and noise. In addition, the low-coordinate space formed by PCA shows some evidence of stability over time.

InteMon [30] is a monitoring system that uses PCA to automatically infer correlations between data streams and alert when those correlations break. The system uses energy thresholding to automatically select the number of dimensions required to represent the data. An anomaly is signaled when the number of dimensions changes, indicating that either a correlation was broken, or a new one was created. Also noteworthy is using the reduced dimensional space obtained by PCA of the original streams to considerably reduce the sensor data storage space requirements. InteMon is a real-time monitoring system, so it uses a stream mining algorithm to perform PCA named SPRINT [31].

2.4.3. User behavior

In [32], Manavoglu et al. present a sequential model for learning individualized behavior models for web users. First, a global behavior model is built for the entire population. Each component of the mixture model represents a dominant pattern in the data, and each sequence (user session) is modeled as a weighted combination of these components. Then, it is personalized by assigning each user individual component weights for the mixture model. They concluded that the Markov model performed better for predicting the behavior of known users, where the maximum entropy (maxent) model was better at the global behavior model (and also unknown users). Note that the maxent model was only computational feasible because of the reduced dimension of the action space.

Mobasher et al. goal in [25] was to discover aggregate usage profiles to use as input source for recommender systems. In order to find overlapping aggregate profiles, two techniques were used: clustering of user transactions and clustering of pageviews. The relevance of this work is the experimental evaluation of the aforementioned profile discovery techniques based on real usage data. The first algorithm, PACT, uses standard clustering algorithms (K-means was used) to partition the space in groups of transactions that are close to each other according to some distance measure. A transaction is an n-dimensional vector in all the pageviews space. The aggregate profiles are generated from the centroids of each group. For each cluster c, a mean vector is computed by finding the ratio of the sum of the pageview weights across transactions in c to the total number of transactions in the cluster. The weights are then normalized, and low-support pageviews filtered out. PACT is similar to method proposed by Karypis and Han in [33], but applied to transactions and not documents.

The second method clusters pageviews across user transactions. The idea is to capture overlapping interests of different types of users. However, since the transactions are now used as features, traditional distance-based techniques cannot be used because there are hundreds of thousands of dimensions. In addition, dimensional-reducing techniques may not be appropriate, as they may discard too much information. Instead, the Association Rule Hypergraph Partitioning (ARHP) [23] technique was used, using the well-know Apriori [34] algorithm.

The authors concluded that PACT was the overall winner in recommendation accuracy, but Hypergraph does better when focused on more restricted data. Hypergraph produced a smaller set

of high quality and more specialized recommendations. On the other hand, PACT has a performance advantage when dealing with all the relevant data.

3 Model and architecture

3.1. Architecture

We followed an incremental approach and opted for a modular architecture. This enables each module to be replaced, upgraded, and even developed without affecting the rest of the system. It also creates a testbed to compare different approaches to solve each of the problems. The architecture is depicted in Figure 9.

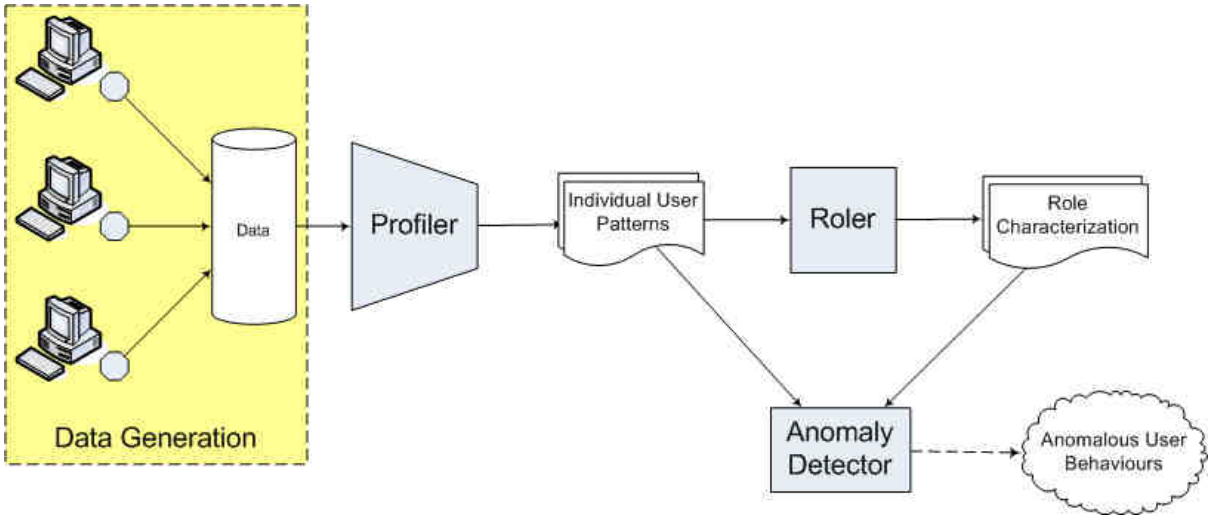

Figure 9 – Architecture block diagram

The first stage, data generation, is executed by Pulso, as described in section 2.1, using the workstation probes and data collection mechanisms. This was the process used to create the dataset. Note that even this part of the process can be replaced with another that produces similar (or even different) data.

The profiler is responsible for interfacing with the data, so it has to execute the pre-filtering operations. Its goal, however, is to characterize each user's behavior according to a model. This tremendously reduces the data size to a manageable level, while remaining useful for the other blocks. Afterwards, the roler will gather all the individual user profiles and group them into roles with similar behavior. Each role will characterize a different type of behavior. The anomaly detector, by comparing the individual user patterns with the available roles will determine which ones have the highest probability of being anomalous behavior. We will now discuss each component in detail.

3.2. Profiler

First, the profiler has to interface with the database where the dataset is stored, adapt and convert the data to a manageable format and pre-filter it. As mentioned before, for this dataset it means to filter out local accounts. Next, the goal is to characterize each user behavior by using tagged flow information. The challenge is how to balance the tradeoffs. On one hand, the profile has to be

specific enough so that the roler can detect similar behavior, and the anomaly detector can detect anomalies. On the other hand, the profile size must be much smaller than original sample size – so that all components can run in acceptable time (i.e., solve the dimensionality problem). In addition, it must scale, so that it can be applied to bigger data samples. The solution is to choose an adequate model that satisfies all these properties. The task of the profiler then becomes to transform the original data into the model format, creating the user profile.

3.3. User profile model

Recalling our dataset information, for each user, there are three attributes of flow information per application: start timestamp, duration and total bytes transferred. Since the innovation of this dataset is the application information, ideally, we would want to characterize the user's behavior using each application. For each relevant attribute, one could capture, for example, the average, the variance and any other number of statistical moments. However, that approach not only throws away too much information, as it is difficult to interpret and explain. For example, average and variance may have some empirical meaning, but what about skewness (asymmetry of the probability distribution), kurtosis ("peakedness" of the probability distribution), and the subsequent statistical moments? We propose a different approach, where we instead use a discretization of the probability density function.

In statistics, the probability distribution (or density) function of a random variable is a function that describes the density of probability at each point in the sample space. The probability of a random variable falling within a given set is given by the integral of its density over the set. We propose a discrete version of this concept, defining a stepwise function with a fixed number of slices. The function is built from the corresponding histogram. This way, we retain most of the sample's relevant features, as a tradeoff with space. The configurable tradeoff parameter is r, the number of slices. Therefore, we construct a discretized probability distribution function (pdf) in the following manner:

1) Define the resolution r – the number of bins (or slices);
2) Split the sample space into sequential non-overlapping intervals, and assign each bin to an interval. Every point in the sample space must be covered by one, and only one bin;
3) All bins are initially empty, and each has an associated counter, with is initialized to zero;
4) For each sample in the set, determine the corresponding bin to which it belongs, and increment its counter value;
5) Normalize each bin counter, so that the total area is 1. In other words, divide the value of each bin counter by the total number of samples across all bins;
6) Define a function pdf(s) that for each input of s, returns the corresponding normalized bin count. s must be in the range {1,2,3,…,r-1,r}.

For example, let's consider total traffic from individual flows. Each sample is the total traffic (in KB) from a distinct flow. The sample set will be {2,6,8,9,11,14,17}, i.e., 7 distinct flows. Let's set

resolution r at r=4. Our sample space belongs to the interval [0,17[, so we'll define the bins interval as b1=[0,5[, b2=[5,10], b3=[10,15[and b4=[15,20[. Note that the bins do not have to be the same size. Any arbitrary bin interval will do, but we will discuss some techniques to dimension them. Applying the rules above, we obtain the results presented in Table 4:

Bin	Interval	Samples	Count	Normalized value
b1	[0,5[	{2}	1	0.14
b2	[5,10[	{6,8,9}	3	0.43
b3	[10,15[	{11,14}	2	0.29
b4	[15,20[	{17}	1	0.14

Table 4 – Calculating a pdf function

The resulting pdf can be represented by the plot depicted in Figure 10. It can be interpreted that most flows are between 5 and 10 KB (corresponding to bin 2), and it resembles a very low-resolution normal distribution.

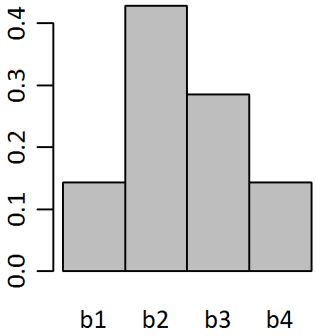

Figure 10 – Example pdf function that uses the normalized values presented in Table 4

Determining PDF slice size

For a given resolution r, there are many ways to define the size of each slice r. To complicate things, most of the attributes sample space is infinite, since it has no upper bound. To solve this, the first step is to limit the sample space. For every period under analysis, there will be a maximum sample value max, such that no sample has a value greater than max. Similarly, there will also be a minimum sample value, typically 0. Now that the sample space is limited to [0, max], the simplest way to define r bins is to make them of equal size max/r, i.e., b1=[0, max/r[, b2=[max/r, 2max/r[, b3=[2max/r, 3max/r[, …, br=[(r-1)max/r, max]. Since the maximum value will be different per application, the slices size will vary per application (but not per user).

This approach is simple but has a major problem: outliers will compress the useful data. Let's consider the example from the previous section and add an outlier to it. The sample set is now {2,6,8,9,11,14,17,1000}, and r=4 as before. For this set, max=1000 and therefore each slice will have size=1000/4=250.

Bin	Interval	Samples	Count	Normalized value
b1	[0,250[	{2,6,8,9,11,14,17}	7	0.875
b2	[250,500[	-	0	0
b3	[500,750[	-	0	0
b4	[750,1000]	{1000}	1	0.125

Table 5 – A pdf function that exhibits compression

As it is clear from Table 5, much information is lost because of the compression effect. We propose a simple solution to avoid this problem. Since, in most distributions, the number of outliers is small when compared to the rest of the samples, we will isolate them in a separate slice. This is accomplished by calculating a high percentile, say 99% (value selected rounded down). The values up to the percentile will be divided in r-1 equally sized slices. The last slice will contain the remaining 1% of the distribution (which will include the outliers, if there are any).

Recalculating the previous example, the 99-percentile of a sample with 8 values would be the 7th value (7,92 rounded down), which is 17. Therefore, each slice will have size=17/(4-1)=5,6. The last slice will contain the values higher than 17. Table 6 presents the results. This simple technique yields much better results, more realistically capturing the distribution of the sample.

Bin	Interval	Samples	Count	Normalized value
b1	[0, 5.6[	{2}	1	0.125
b2	[5.6, 11.3[	{6,8,9,11}	4	0.5
b3	[11.3, 17.0[	{14}	1	0.125
b4	[17, +∞]	{17,1000}	2	0.25

Table 6 – A pdf function calculated using a percentile

Determining attributes

Of all the attributes of the dataset, the minimum set that allows characterizing behavior should be selected. The applications each user uses is certainly important to characterize his behavior. The per application flow attributes (traffic per flow, flow duration, and flow start time) seem good candidates at discriminating user behavior. In addition, they are numeric attributes, which is the type best suited for the kind of analysis that we want to perform.

The remaining attributes, destination IP and port are highly correlated to the application, not bringing any new information. In addition, they are categorical attributes, which would increase the dimensionality too much (because of the unfolding, more on this on the next section).

In conclusion, for each application, we define three pdf functions that consider flow counts for each of the following attributes: traffic per flow, duration per flow and hour of day. The model of the user behavior is the set <application, traffic pdf, duration pdf, time_of_day pdf>, for every application used in the considered period.

3.4. Roler

The goal of the roler is to group similar user behavior into groups or roles[1] and to characterize them. It receives as inputs all the profiles, which are instances of the model described in section 3.3. For this kind of analysis, the most promising data mining technique is clustering. Anomaly detection is obviously not what we want (at this stage). Classifiers are a possibility, since they build a model – the ones that build black box models would be excluded. However, their main goal is to label samples with a certain class, as learned from the training set. In our case, we do not have a definite set of classes we want to distinguish. We want to find out natural, similar behavior. Because of this, they are inadequate. Association analysis is more geared towards discovering relationships of the type {Firefox} -> {Word}, that is, Firefox users are likely to also be Word users. While interesting, it is not what we want. Also, like mentioned in section 2.3.5, association analysis typically uses categorical attributes, and discards the number of uses. Finally, the goal of clustering is finding similar objects and group them together. Because of this, clustering techniques are the best candidates for this component.

However, clustering can only be applied to numerical data. Consider the model we have: each user profile is a set of <application, traffic pdf, duration pdf, time_of_day pdf>, application is a categorical attribute, and each pdf is a structured attribute. This can be solved by applying dimensional unfolding, which consists in transforming every combination of categorical attributes into new dimensions.

For our particular model, this occurs as follows. The starting point is a pdf slice with an associated numerical value, the expected probability of a flow occurring in that slice's range. Each pdf has r slices. Each user profile has three pdfs, one for traffic, one for duration and other for time of day. All these 3*r slices will be distinct dimensions. The profile has these three pdfs for each application used. Each application will create new 3*r dimensions for its pdfs. In other words, each dimension in our model is a <application, attribute, pdf slice number> tuple. Table 7 exemplifies the unfolding process for our model.

Dimension	Value
<Firefox, traffic, slice 1>	0.80
<Firefox, traffic, slice 2>	0.10
<Firefox, traffic, slice 3>	0.05
<Firefox, traffic, slice 4>	0.05
<Firefox, duration, slice 1>	0.30
(…)	
<Firefox, duration, slice 4>	0.10
<Firefox, time_of_day, slice 1>	0.00
(…)	
<Firefox, time_of_day, slice 4>	0.10
<Word, traffic, slice 1>	0.60
(…)	

Table 7 – Dimensional unfolding example (k>=2, r=4, three attributes)

[1] Throughout this document, the terms group, cluster and role are used interchangeably, referring to the same concept.

This solution creates a high dimensional space, since for k applications used, for r slices per pdf and for our three attributes, the number of dimensions will be k*3*r. The input data for the clustering algorithm is conceptually a matrix, with m users (rows) and k*3*r dimensions (columns). Therefore, a suitable clustering algorithm must be used.

K-means is simple, effective and fast (linear time and storage is just samples*features). However, as with all clustering algorithms, the issue is in determining the right parameters (at least initially). In K-means, the parameter is K, the number of desired clusters. The problem is we have no apriori knowledge of how many clusters are adequate. The typical solution is to iteratively run K-means with increasing values of K, and calculate the total sum of squared error. When the error variation is below a certain threshold, K is selected.

X-means is an improvement of K-means, since it automatically determines K in a much more time-efficient manner, by experimenting which clusters are best to split. It is also much more memory efficient, since it uses kd-trees to store the data. For these reasons, X-means was the clustering algorithm used in the roler.

3.5. Roles model

The clustering algorithm will calculate a centroid for each cluster, the means of each dimension. Since each dimension is a slice of a pdf, each cluster centroid will be a set of pds. Therefore, the centroid pfds are the model for the corresponding cluster (role). This pattern describes the role (group) behavior.

In effect, it is a pattern just like the user patterns (with the same dimensions), but describing the entire group. Nevertheless, since each dimension is calculated independently, what guarantee is there about the area of each pdf? We will show that if all input pdfs have unit area, each resulting centroid pdf also have unit area.

Theorem 1: If all input pdfs have unit area, each resulting centroid pdf will also have unit area.

Proof: A function pdf(s) is defined as in section 3.3, i.e., the probability that a given value lies in bin s.

A function pdf(s,u) is a function that returns pdf(s) for the corresponding user u. All pdfs have the same total number of bins r. By construction, these functions have a total area of 1, i.e.,

$$\text{A1)} \sum_{i=1}^{r} pdf(i,u) = 1$$

The clustering algorithm will calculate each pdf slice independently, where each slice s is the average of the slice $s \in \{1,2,3...,r\}$ of each user u in cluster c.

$$\text{A2)}\ pdf(s,c) = \frac{\sum_{j \in c} pdf(s,j)}{|c|}$$

The total area of the pdf function for cluster c is given by

$$\text{A3)} \sum_{i=1}^{r} pdf(i,c)$$

Which, using A2) is equivalent to

$$\text{A4)} \sum_{i=1}^{r} pdf(i,c) = \sum_{i=1}^{r} \frac{\sum_{j \in c} pdf(i,j)}{|c|} = \frac{\sum_{i=1}^{r}\sum_{j \in c} pdf(i,j)}{|c|} = \frac{\sum_{j \in c}\sum_{i=1}^{r} pdf(i,j)}{|c|}$$

According to A1)

$$\text{A5)} \ \frac{\sum_{j \in c}\sum_{i=1}^{r} pdf(i,j)}{|c|} = \frac{\sum_{j \in c} 1}{|c|} = 1$$

■

However, not every user uses every application, which means that, in a user profile, there will be pdfs with unit area, corresponding to the used applications, and pdfs with area 0, corresponding to the not used applications. In this case, we will show that each resulting centroid pdf area equals the ratio of users (in that cluster) that used that application over total users of that cluster.

Theorem 2: If all input pdfs have either unit area or area equal to zero, each resulting centroid pdf area will equal the ratio of users (in that cluster) that used that application over total users of that cluster.

Proof: Like before, function pdf(s) is defined as in section 3.3, i.e., the probability that a given value lies in bin s.

A function pdf(s,u) is a function that returns pdf(s) for the corresponding user u. All pdfs have the same total number of bins r. However, now, some pdf functions have unit area, just like A1:

$$\text{B1)} \sum_{i=1}^{r} pdf(i,u) = 1$$

Others have area 0, corresponding to the applications the user did not use:

$$\text{B2)} \sum_{i=1}^{r} pdf(i,u) = 0$$

Also as before, the clustering algorithm will calculate each pdf slice independently, where each slice s is the average of the slice $s \in \{1,2,3...,r\}$ of each user u in cluster c.

$$\text{B3)} \ pdf(s,c) = \frac{\sum_{j \in c} pdf(s,j)}{|c|}$$

Again, the total area of the pdf function for cluster c is given by

$$\text{B4)} \sum_{i=1}^{r} pdf(i,c)$$

Which, according to A4 is equivalent to

$$\text{B5)} \sum_{i=1}^{r} pdf(i,c) = \frac{\sum_{j \in c} \sum_{i=1}^{r} pdf(i,j)}{|c|}$$

Let's divide cluster c into two sub-clusters: c_0 containing the users that did not use the application (rule B2) and c_1 containing the users that did use the application (rule B1). B5 can be rewritten as

$$\text{B6)} \sum_{i=1}^{r} pdf(i,c) = \frac{(\sum_{j \in c_0} \sum_{i=1}^{r} pdf(i,j)) + (\sum_{j \in c_1} \sum_{i=1}^{r} pdf(i,j))}{|c|}$$

By substituting the first part with B2 and the second with B1, we have

$$\sum_{i=1}^{r} pdf(i,c) = \frac{(\sum_{j \in c_0} 0) + (\sum_{j \in c_1} 1)}{|c|} = \frac{|c_1|}{|c|}$$

■

These properties mean the model is correct, in the sense that the resulting pdfs represent the behavior of most users, and also yields information about the 'popularity' of each application in each cluster. For example, if cluster 4 has a pdf for application A with area of 1, that means that all users in that cluster used it. On the other hand, if cluster 5 has a pdf for application B that only has an area of 0.5, only half of the users in that cluster used it.

Application energy

In a profile, every application is characterized by 3 pdfs, one per attribute (traffic, duration, time_of_day). When a user utilizes that application, all pdfs have area 1, and when he does not, all three have area zero. As it was shown before, each application pdf in a cluster centroid will have an area proportional to its cluster usage ratio. This means that the area for any of the three pdfs that characterize it will be the same. Let's define the average of the 3 pdfs area from application X as its 'energy'. Since all three areas are the same, the application energy will also be equal to any of its three pdfs area. The application energy will be in the range [0,1], since this is the range for each pdf.

This property reflects the importance of each application in a cluster. High-energy applications are the most relevant for that cluster, and characterize it. The shape of each pdf characterizes how the users in that cluster used that application. For example, the same high-energy application in different clusters might be used differently, thus, the pdfs will have different shapes.

3.6. Anomaly detector

As seen in section 2.3.4, the are three main types of anomaly detection techniques: model-based, proximity-based and density-based. Since we already have a model for the users and for the roles, the model-based technique is the most cost-effective to use.

The main idea is that users that are farthest from clusters centroids are anomalies. For each user profile, we define an anomaly score as the sum squared error to the nearest cluster centroid. We calculate the anomaly score for each user, and sort them by that score. The highest scores up to a threshold or the top x% are considered anomalies.

In the complete process, each anomaly may be manually analyzed, comparing the applications used with the ones on the nearest cluster. Sorting the applications by total error will yield the ones that were the most 'anomalous'. For popular applications, a side-by-side comparison of the pdfs may be useful, revealing the differences in behavior.

An additional step is required, though. An entire group may be "anomalous", in the sense that it exhibits strange behavior. The elements of these groups will be near their centroids, and will not be flagged as anomalies. The only way to detect them is through manual inspection of each group profile, a step that was already recommended.

3.7. Temporal analysis

The architecture we have seen so far can be applied to any period: day, week, month, year, etc. Nevertheless, it is a single analysis. By making multiple consecutive analyses and comparing them, one could examine the evolution of:

- Anomaly scores per user. The case of a user that exhibited anomalous behavior in just one period is different from the case of a user that has the said behavior for some periods. This analysis also allows making a distinction between single occurrences (spikes), and persistent behavior. It will also allow verifying if the anomaly score for a user is stable, increasing or decreasing;

- Cluster stability. This analysis will show how the roles are evolving over time, and if they are stable or show much variation.

3.8. Previous models

The model described so far is the final version, and the culmination of this research project. It was neither the first nor the only one tried. The lessons learned from the previous models that lead to the success of the final one are also valuable. Therefore, this section briefly analyses the early approaches at solving the problem.

3.8.1. PCA / SVD

The first attempt was to use principal component analysis (PCA) and similar techniques (like singular vector decomposition) to perform dimension reduction on the dataset. The idea was to discover which applications were important to tell users apart. The dataset was unfolded into application-attribute pairs, but only a single attribute at the time. Only one value was considered by attribute, usually the sum. The resulting scree plot revealed a heavy-tailed distribution, meaning that many dimensions were required for accurate separation. Not only that, but the principal vectors contained contributions from practically all the applications, making it difficult to distinguish the important ones.

The analysis was interesting, and helped shed some light into the dataset. Nevertheless, this line of research ultimately showed no useful results, and was abandoned to focus on more domain-specific approaches.

3.8.2. Overly simplistic model

The first domain-specific model was very simplistic on purpose, to get a more detailed feel of the data. Of course, had it sufficed, this would have been a much shorter project. The application information was ignored on purpose, and a tridimensional space was defined: <duration, time of day, total traffic>. Every flow was plotted on this tridimensional space, represented by a dot in Figure 11. All the dots have vertical lines connecting them to the XY plane. The X-axis (horizontal) represents duration, the Y-axis (depth) is the time of day and the Z-axis (vertical) is traffic.

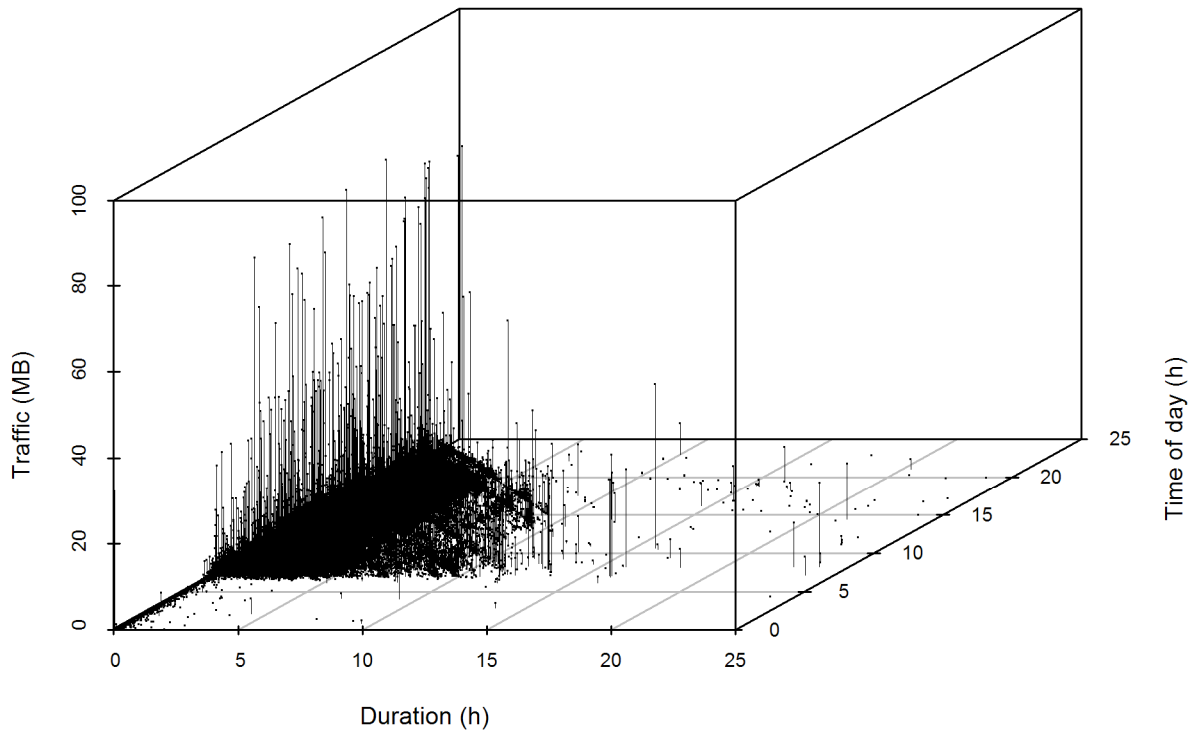

Figure 11 – Tridimensional (duration, time, traffic) plot of dataset

There are no evident groups or separations, as expected. Clustering also did not yield any meaningful separation of the three classes (operators, supervisors, others).

These were also the first attempts at developing visualization techniques for the dataset. We agree that the human visual system is a "pattern seeker of enormous power and subtlety" [35]. Since this system is to be used by human operators, it makes sense to display the data in the most meaningful manner.

3.8.3. Fixed categories on individual attributes (traffic/flows/duration)

On this model, each application is a dimension. A single flow attribute is used (traffic, number of flows or duration). For each user flow, the values of the selected attribute are added to the corresponding application. In the end, the values are normalized. In effect, this determines the applications each user uses the most, according to the selected flow attribute.

Experiments were made for each of the three attributes, and the results were similar. The clusters showed an even distribution across all three classes. No natural groups could be discerned.

3.8.4. Variable categories on all attributes

This model is an improvement over the previous one. The idea is to generate more dimensions, splitting each application into a set of usage patterns. It is a two-step model:

First, cluster each application individually, to determine different application usage patterns. For each application, the entire dataset was used (all users). The space had three dimensions (time of day, duration, and traffic). These clusters defined usage categories.

Second, using the categories from the previous step, classify each user flow, like in the previous model. In the end, normalize the matrix.

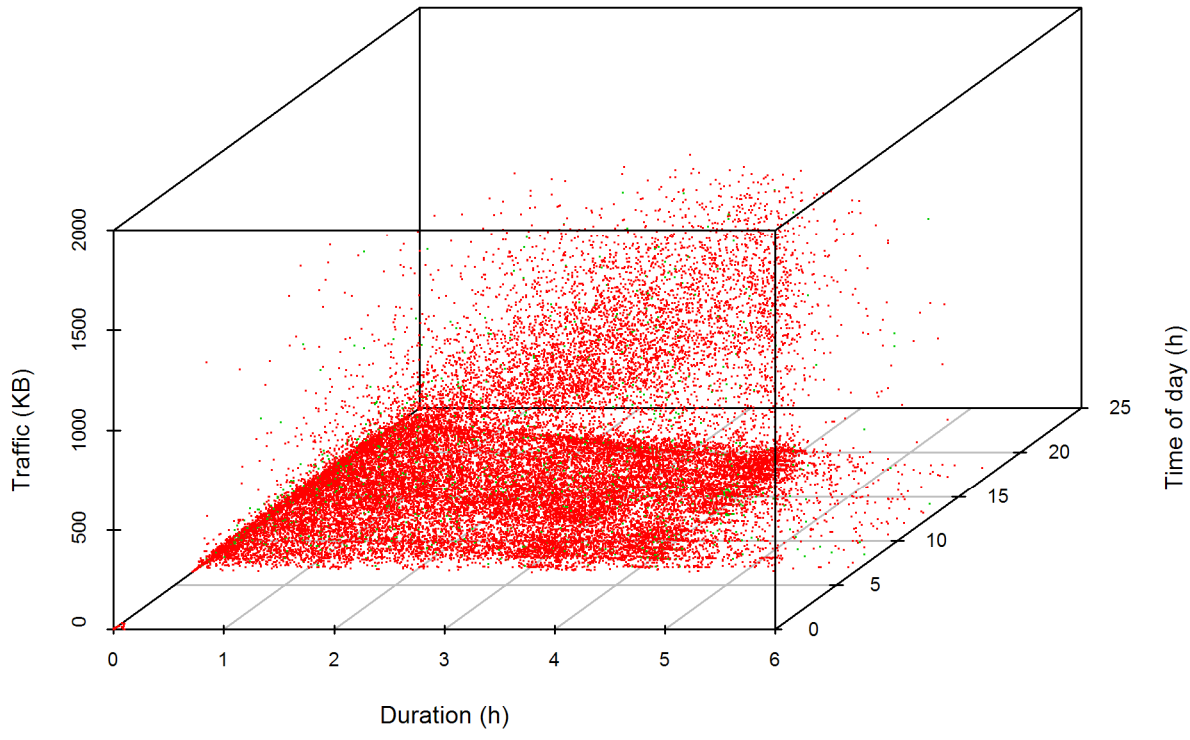

Figure 12 – Scatterplot for application 520

Again, the results were not satisfactory. To find out why, scatter plots like the one in Figure 12 were generated for each application. Each dot represents a flow in the <duration, time of day, total traffic> space. The colors (if available) indicate the class: red is class 0, green is class 1 and blue is class 2 – they are not that important, since class 0 dominates the plot, and the remaining classes can barely be seen. The X-axis (horizontal) represents duration, the Y-axis (depth) is the time of day and the Z-axis (vertical) is traffic.

From the manual analysis, the conclusions were:

- The categorization step only creates categories if a majority of the users behaves differently. Therefore, distinct but minority behavior will not be analyzed;
- By using that few categories, many important information from the dataset is lost. Therefore, the results are poor;
- Since the results are normalized, if a user behaves like all others, but uses a single application that unbalances the analyzed attribute (like much traffic or many flows), the relative percentage of the rest will decrease, making the user too different. This yields poor results;
- On the other hand, if the results are not normalized, it is difficult to compare different users, because of their different workloads;
- Treating each application individually was biasing the results, and the user separation process.

The final model handles these issues by using more information (one pdf with r slices per attribute) and only one grouping phase, to pickup subtle differences in behavior.

4 Results

A prototype of the proposed system described in Chapter 3 was implemented using a combination of Java and the R statistics program [36]. JRI in the rJava package [37] allowed Java to interface with R. This chapter presents and discusses the results.

4.1. Single analysis

The longest time frame on our dataset that had some period cycle was one month. Therefore, the first month of data was analyzed. Analysis on the second month yielded similar results. This period, corresponding to the second month, has 841 users and 203 distinct applications. The distribution for this period is similar to the one for the entire dataset:

- Class 0 (Operators): 765/841 (91%)
- Class 1 (Supervisors): 48/841 (5,7%)
- Class 2 (Other): 28/841 (3,3%)

Running the Profiler, we obtain the user profiles. Each user profile is a set of pdfs for the applications used by that particular user. Figure 13 contains the three pdfs for user 2 and application 58.

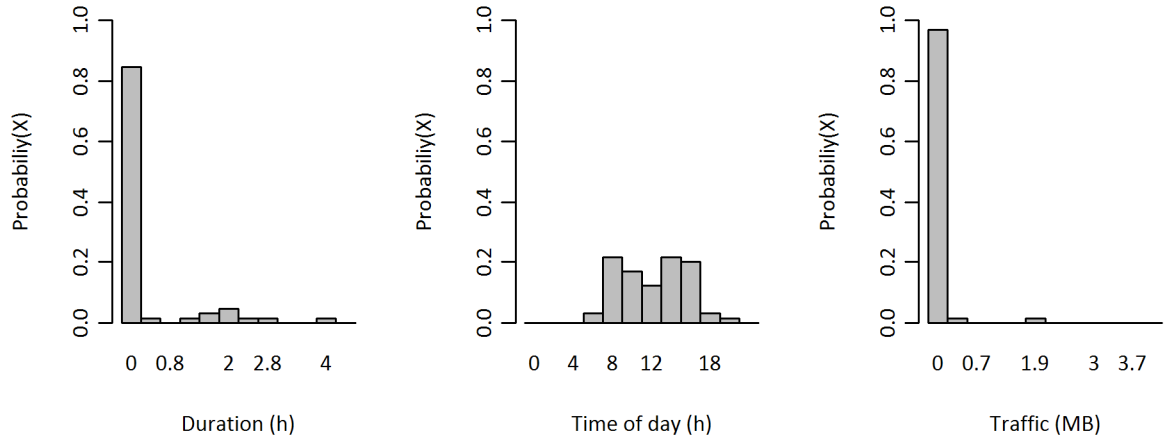

Figure 13 – PDFs for user 2, application 58

From left to right, the duration slices have width=0.4 hours, the time of day slices have width=2h, and the traffic bins have width=400KB. The application usage can be interpreted as containing mostly short duration flows. The application is used during the work hours, since the time of day exhibits the typical two hump daily distribution. Most of the flows contain little traffic. Now, if we unblind application 58 and reveal it is a web browser, this means the user mostly navigates in webpages during working hours. Let's compare this, with the pattern of user 252, in Figure 14.

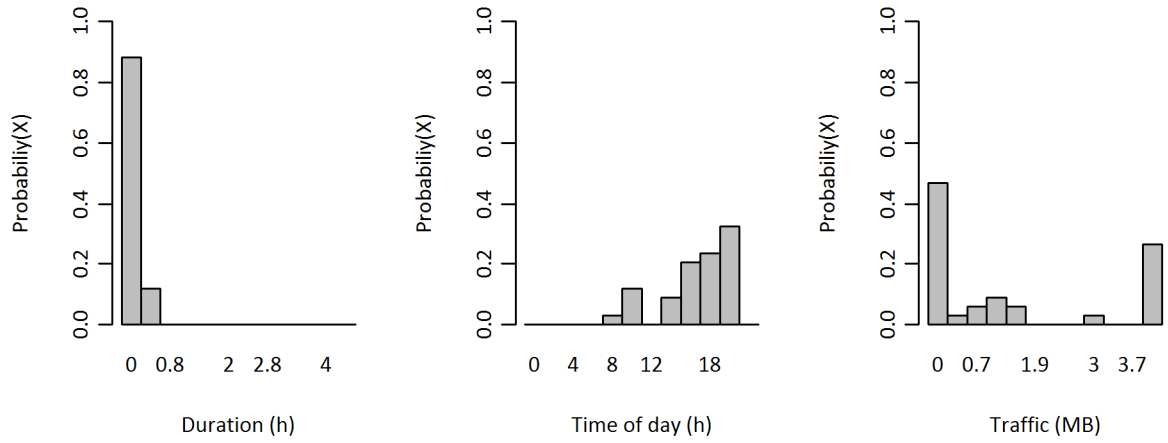

Figure 14 – PDFs for user 252, application 58

These pdfs show a much more skewed time-of-day distribution, somewhat longer flows, and distributed between light and heavy traffic flows. Probably this user has to download large files on a regular basis, leaving them downloading after work hours

This is an example of what the user model can characterize, per application. The profiler produces these distributions for each application, for each user. This multi-dimensional dataset can be represented as a matrix. We devised a visualization method for this data, designated 'user/application spectrogram', a bird's-eye view on all the user profiles.

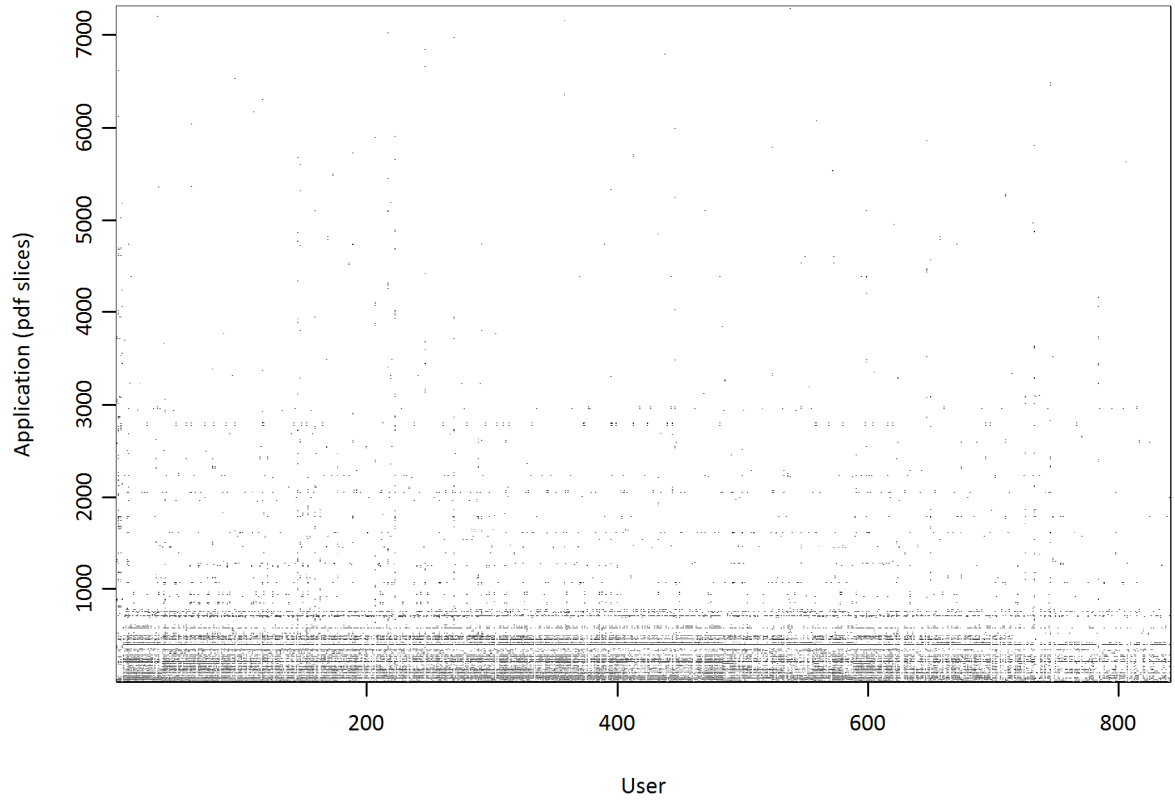

Figure 15 – User/application spectrogram (high contrast)

In a user/application spectrogram, each column corresponds to a user profile. Each consecutive set of rows will contain one application vector. Each row contains one dimension, unfolded as explained in section 3.4. It can be seen as if the three pdfs values are stored in a vector corresponding to an application. Therefore, each row contains one of the slices of the corresponding application-attribute pdf. Each value in this matrix is the value corresponding to the pdf of user x, and application-attribute-slice y. Using a black-and-white palette, black corresponds to value 1 and white to value 0, we obtain the User/Application spectrogram in Figure 15. We present the high contrast versions of the spectrogram to better show its features. The contrast increase was obtained by applying a non-linear function to the (linear) tone scale. The function used was $f(x)=x^a$ with $a<1$.

The applications were ordered so that the most used ones appear at the bottom. Interestingly, despite the human ability to detect patterns, there does not seem to be any discernible ones. However, after running the roler, nine roles are defined, shown in Figure 16. The vertical (blue) dashed lines indicate cluster separation, and the users were re-ordered so that the clusters are contiguous.

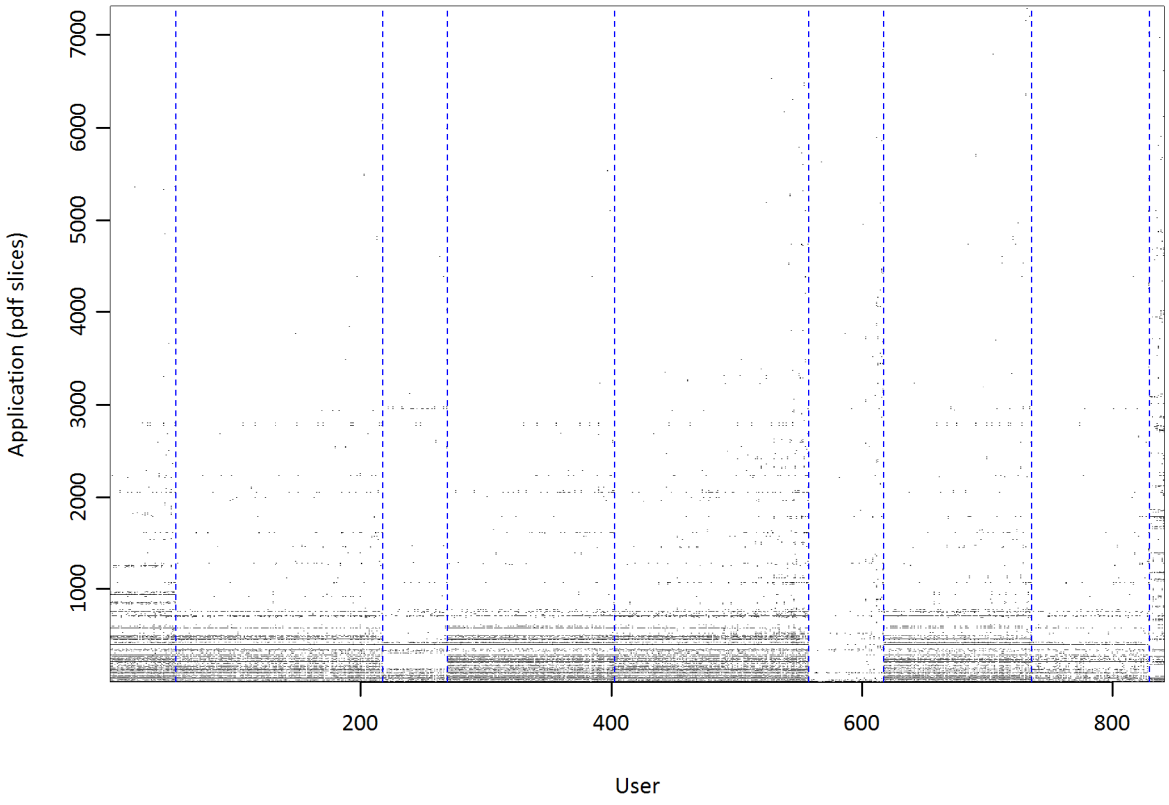

Figure 16 – User/application spectrogram (high contrast) after clustering

The inter-group differences and the intra-group similarities can clearly be seen. Figure 17 is a detail of the bottom part of Figure 16. It contains the 25 applications with highest number of flows. Horizontal (blue) dashed lines were added to separate distinct applications.

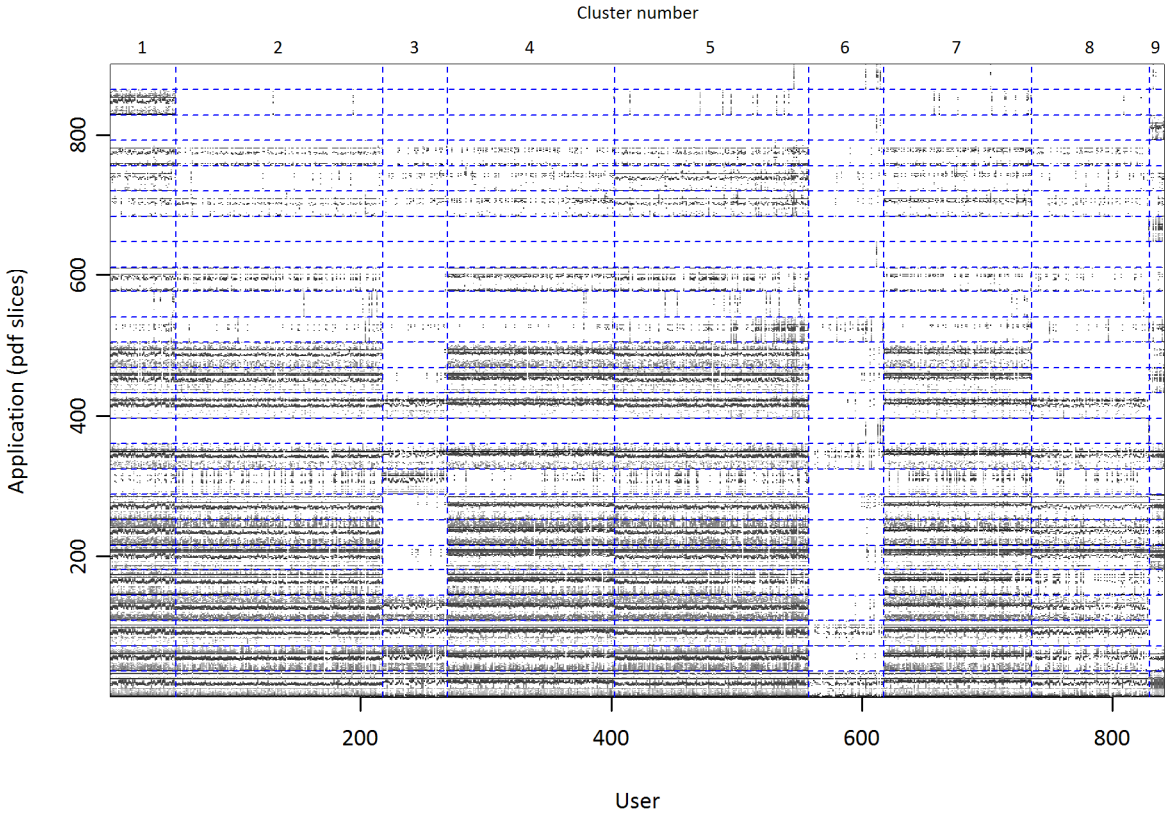

Figure 17 – User/application spectrogram (high contrast) detail after clustering

Now, humans can visually evaluate similarities and differences in quick and effective way, which is the strength of the User/Application Spectrogram. Let's number the clusters from left to right, 1 to 9. The class/role distribution is presented in Table 8.

Role	Class 0 (Operators)	Class 1 (Supervisors)	Class 2 (Others)
1	**49**	3	0
2	**164**	1	0
3	**52**	0	0
4	**132**	1	0
5	137	**18**	0
6	35	**10**	**15**
7	**114**	4	0
8	82	**11**	1
9	0	0	**12**

Table 8 – Role user distribution per class

From the data in Table 8, several aspects can be concluded. First, supervisors do not seem that different from operators, they appear mixed with them. Some roles contain mostly operators, namely roles 1 through 4 and 7. The supervisors are mixed with operators mostly in roles 5, 6 and 8. There is even an entire role (role 9) with just users from class 2, the 'Others' group. This role is clearly different from the others, as can be seen from the spectrogram. Role 3 is also different from the rest, and contains mostly operators, which is quite strange. Role 6 contains a mix of every class. From the spectrogram, roles 3 and 6 also stand out, backing the results on the table.

4.1.1. Role visualization

Similar to the user profiles, each role is also characterized a set of applications and their behavior (pdf function) in each of the three attributes (traffic, duration, time). As an example, one of the most used applications for role 3 – application 476 – can be seen in Figure 18.

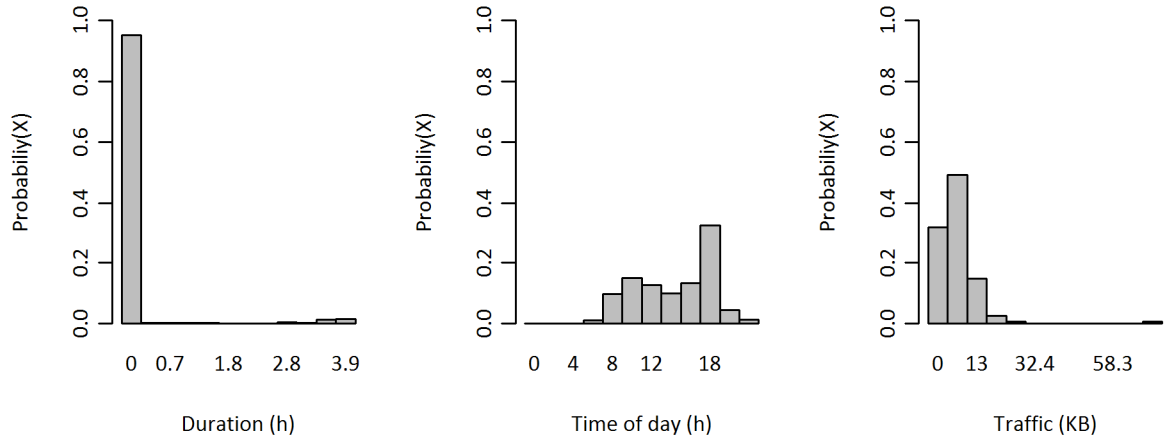

Figure 18 – Application 476 in cluster 3

This application is characterized by fast interactions (low duration flows), and light traffic (around 6KB per flow). It shows the usual two hump cycle, but there are many interaction between 18-20h, probably due to functions performed at office closing hours. This type of characterization is exact, but lengthy, since there are many applications.

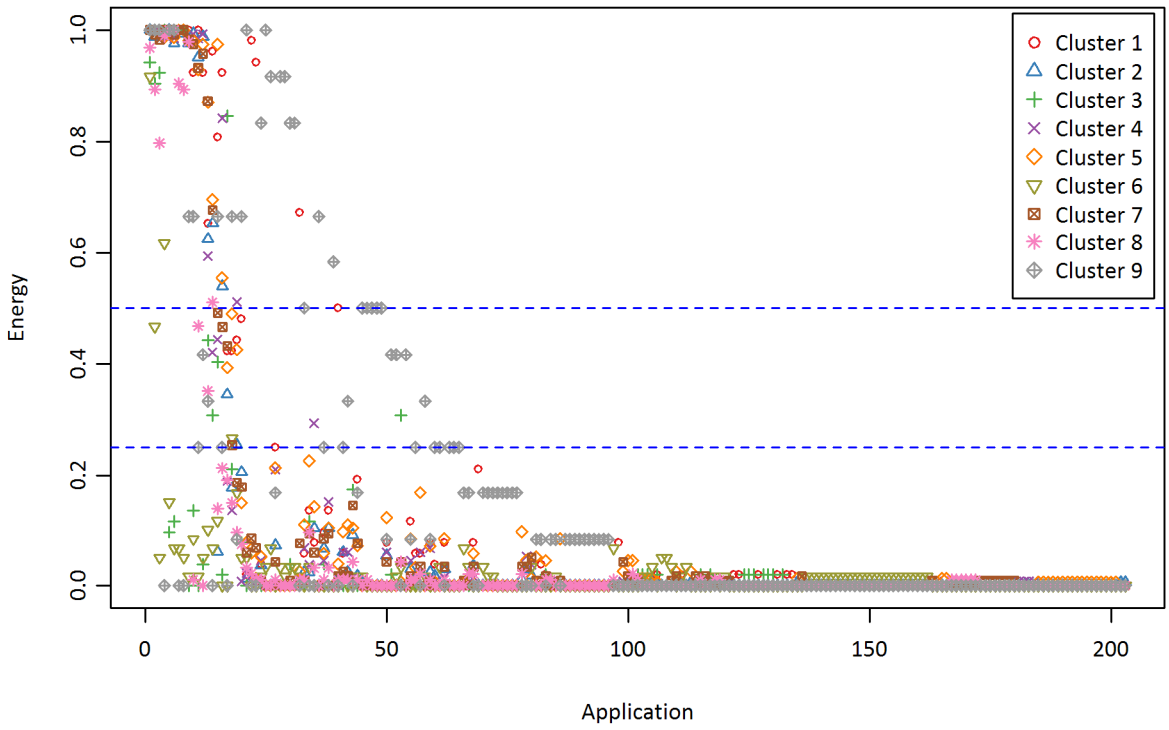

Figure 19 – Application energy plot for all clusters

To solve this, we developed a visualization that allows quick role characterization: by using the energy per application (as defined in section 3.5, page 30), it can be seen which applications each roles use most. Designated 'Application energy plot', this visualization plots the energy for each application. An example for our nine roles can be seen in Figure 19. The dashed lines indicate the 50% and 25% borders. It can be seen that each role has its distinct signature.

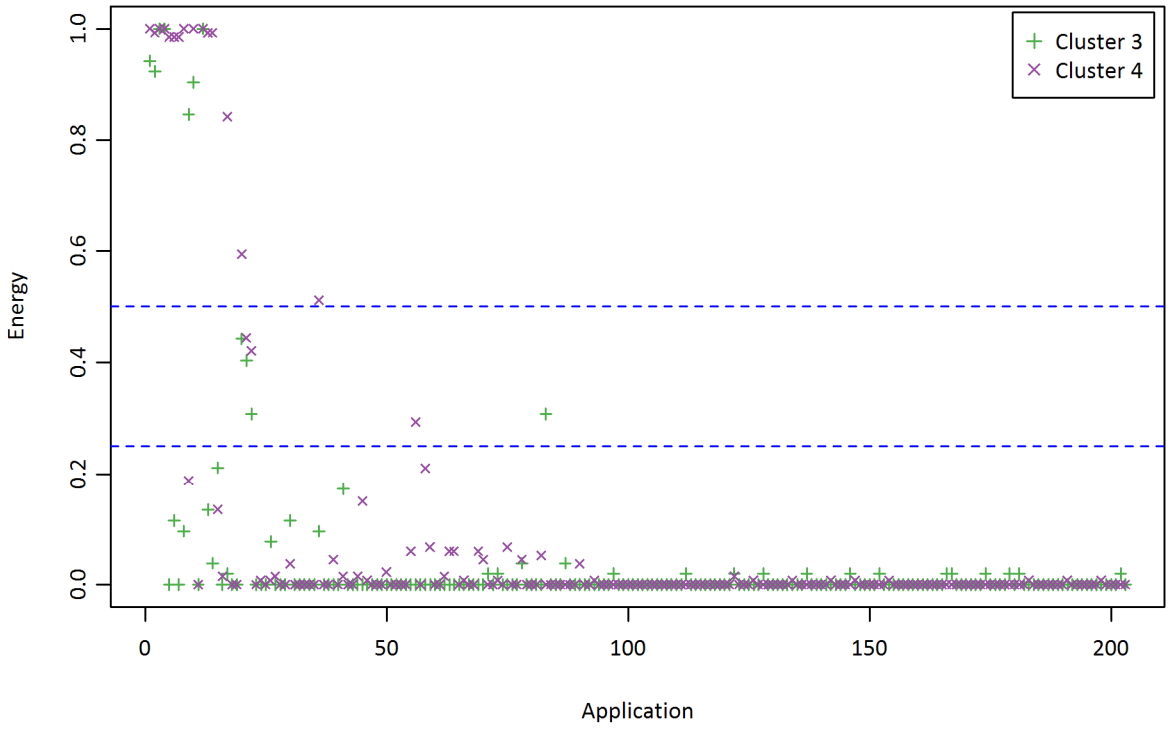

Figure 20 – Application energy plot for clusters 3 and 4

Cluster 3	
Usage	**Application**
100%	Business 533
100%	Business 520
100%	Business 476
94%	Login 6
92%	Business 236
90%	Web Browser 58
85%	**Business 552**

Cluster 4	
Usage	**Application**
100%	Business 533
100%	Business 520
100%	Business 476
100%	Login 6
100%	**Business 267**
100%	Web Browser 58
99%	**Business 275**
99%	Business 236
99%	**Business 330**
98%	**Business 319**
98%	**Business 105**
98%	**Business 131**
84%	**Business 218**
59%	**Office 133**
51%	**Office 264**

Table 9 – Cluster 3 vs cluster 4 applications breakdown (over 50%)

Let's compare roles 3 and 4 seen before on Figure 17. For that, let's use the Application energy plot for just roles 3 and 4 in Figure 20. The differences between which applications each role uses the most can be seen. There are some common applications on the upper left quadrant, but each series has a distinct signature. Table 9 shows a more detailed breakdown (for applications used by more than 50% of the role) with differences highlighted in bold.

Role 3 has a unique application when compared to role 4 – Business 552. Role 4 has many more unique applications, including office applications. As expected, there is a common set of applications (after all, it is the same general population). This information confirms and explains the differences seen in the spectrogram detail (Figure 17, page 38). Role 3 has many more white areas because there are a number of applications that are not much used when compared with role 4.

Nevertheless, even applications used by many users can be used differently. Let's compare Business Application 476 in Figure 21 (role 3) with Figure 22 (role 4). Both have the same short duration flows (less than 18 minutes). However, role 3 uses it throughout the day, with greater emphasis between 18-20h. In addition, there is a greater variability on flow traffic, maybe due to a more diverse set of operations. Cluster 4 only uses it from 14h-22h, potentially an afternoon shift. In addition, traffic flow is more regular, between 6KB and 13KB, pointing towards a more stable set of operations.

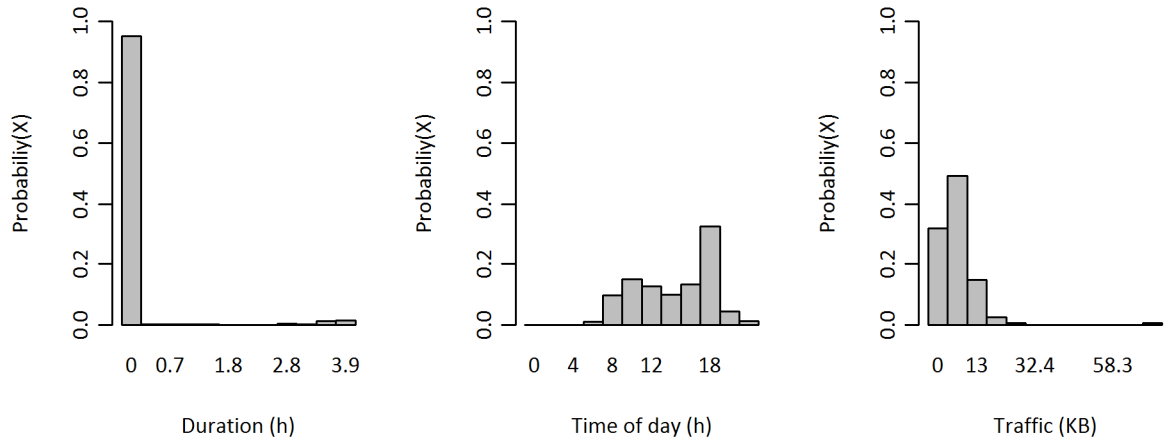

Figure 21 – Business application 476 in cluster 3

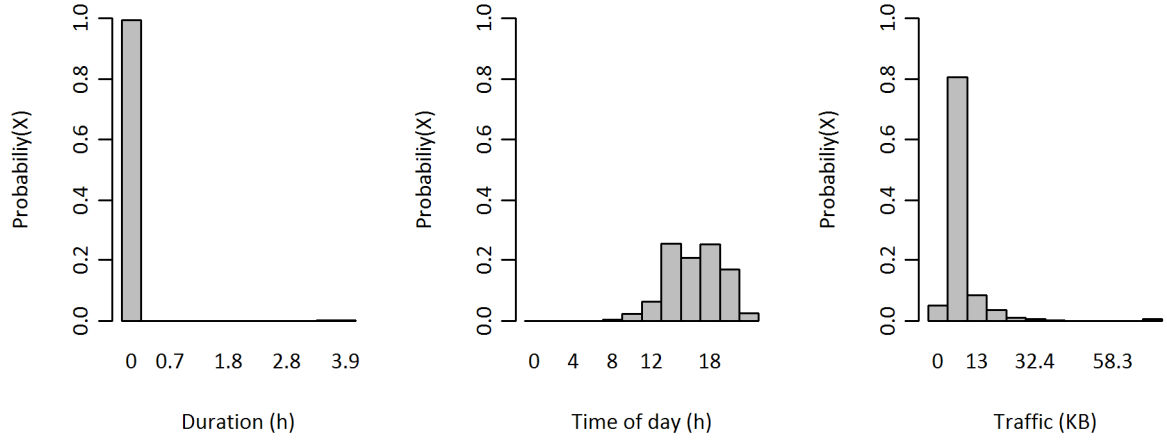

Figure 22 – Business application 476 in cluster 4

4.1.2. Anomaly detector

We have seen how our model can characterize individual user behavior, role behavior and how visualizations can be used to quickly inspect data. The final component of the system is the Anomaly Detector. It computes an anomaly score for each user, the 'distance' to the nearest cluster centroid. To visualize these scores, the 'Anomaly Score' plot contains the sorted scores for all the users.

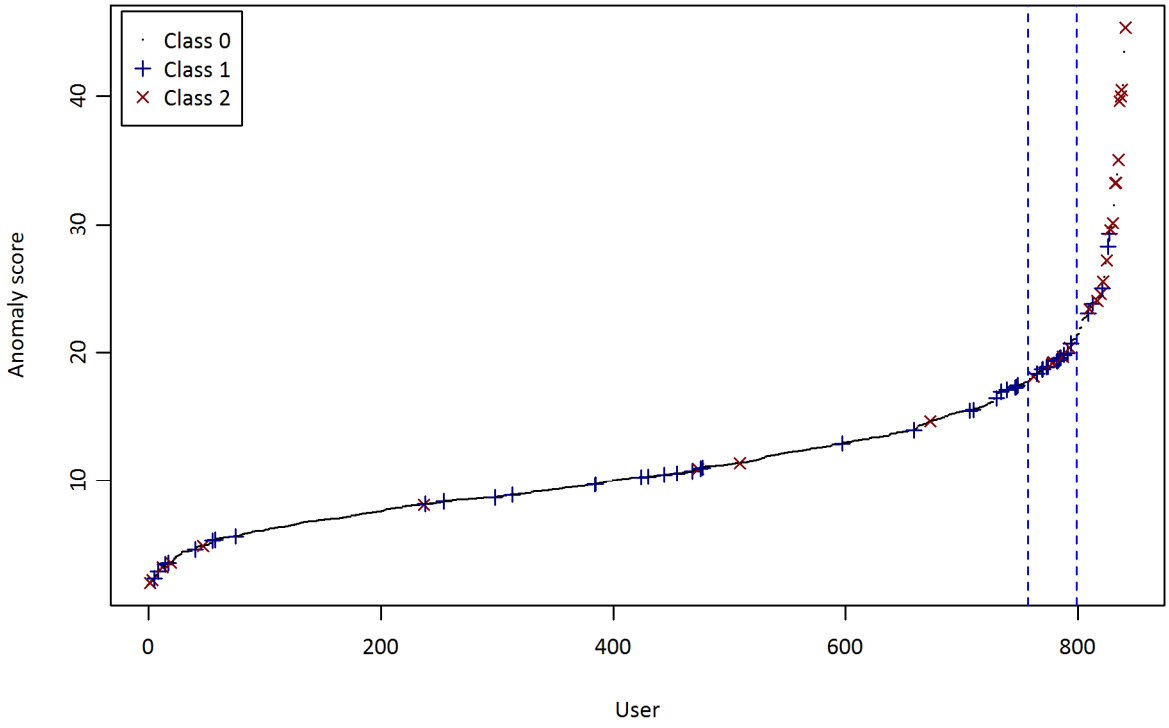

Figure 23 – Anomaly score plot

The X-axis has the users sorted by anomaly score, and the Y-Axis, the anomaly score. The vertical blue dashed lines separate the top 10% and 5% of the users. Each class is plotted with a different symbol and color.

Based on our knowledge of the dataset we were expecting class 2 to be mostly anomalies, since they have great freedom and privileges. We were also expecting for class 1 to be anomalies, since they have much greater freedom than operators do. The operators (class 0) should all be similar and 'well-behaved', not anomalies.

Therefore, we are expecting class 2 and class 1 to be most of the anomalies. To evaluate the results we also have to take into account the base distribution. Recalling, the base distribution is:

- Class 0 (Operators): 765/841 (91%)
- Class 1 (Supervisors): 48/841 (5,7%)
- Class 2 (Other): 28/841 (3,3%)

Class 1 and 2 together represent only 9% of the total samples. Therefore, at least 9% of the anomalies the detector flags should be either class 1 or class 2 samples. By comparing our results

against these 9% we get an estimate on how good the anomaly detection is. The anomaly detector had the results shown in Table 10, confirming the data on the anomaly score plot. 70% of the top 1% higher anomaly score were either class 1 or 2, a 7.8 times increase over the base distribution (9%). For the top 5% and 10%, more than 40% of the anomalies were class 1 or 2. This is a significant increase (over four times) over the base distribution. These results were what we expected. Let's analyze the top two anomalies shown in the plot.

Anomalies	Class 1+2	Increase over base distribution (9%)
Top 1%	70%	7,8
Top 5%	43%	4,7
Top 10%	42%	4,6

Table 10 – Anomaly detection results

Class 2 anomaly

The anomaly with the highest score is user number 2, and as it can be seen from Figure 23, belongs to class 2 (top right red x mark). Therefore, his role uses many and uncommon applications. But even considering those, he uses 72 distinct applications, most of which no one else on his role uses. This diversity accounts for the bulk of his anomaly score.

Manual and individual inspection of the applications reveals that many are not business-related, some require administrator privileges, and some are unknown to us. Clearly, either this user is cleared for such behavior or it is in violation of the security policy. It is, in fact, an anomaly, that warrants further investigation.

Class 0 anomaly

These anomalies are much more promising and interesting, since one would expect all class 0 users to be operators and behave in a similar fashion. The user in question is user 10, belongs to class 0 (operator), cluster 5. Recalling from Table 8, cluster 5 contains mostly operators, and some supervisors.

The user had 90 distinct applications. A quick comparison of their names to the ones on his cluster reveals a great mismatch. We identified media playing applications, network debug tools, and different business applications. It is very suspicious that a callcenter operator would use such applications. It also warrants further investigation.

During this analysis, we realized that on screen side-by-side comparison of the user's pdfs and the cluster's pdfs for the applications is a useful technique to spot subtle differences.

On both analyses, we omitted the application list table versus cluster. Since the application names are blinded, it would not be much useful to display a long table of 'number applications'.

5 Deployment

This section details the major concerns in deploying this system. It also analyses issues relevant for deployment in a corporate environment.

5.1. System components

5.1.1. Probes

Although not part of this project, the probes must be deployed on workstations to generate the relevant events for this system. Since most corporations have control over the images on the workstations, and the deployment process, this should be easy to accomplish. The probe should generate little traffic (as not to be intrusive) and stable.

The per-workstation traffic volume is already low, but sampling can be used. It will be most effective if applications with many volumes are sampled, instead of randomly sampling flows (since it might affect applications with little flows). Sampling can also be done at the workstation level, which means not all users will be monitored. This may affect the results, because some potential outliers may be missed. It is recommend that all users are monitored, especially since the probes are light, unobtrusive and stable.

The events that are sent from the probe to the central event repository should be protected, depending on the attacker model:

- If an attacker has eavesdropping capabilities, the confidentiality of the events should be assured. Otherwise, the attacker will gain information about what applications each user utilizes;
- If an attacker has injection capabilities, the authenticity of the sender should be assured. Otherwise, the attacker can poison the database with false events. The authenticity of the event repository should also be assured; otherwise, an attacker may impersonate it. Freshness mechanisms (like timestamps and/or nonces) should be used to prevent replay attacks;
- If an attacker has modification capabilities, the integrity of the message should be assured, otherwise, the attacker could mask anomalous behavior.

5.1.2. Event repository

A listener will store the events in a backend database. The listener should be light to reduce latency. The database will be queried by the profiler to provide data. The datasets will most likely not fit in memory, so disk throughput is the main concern. It will have very little CPU requirements. Concurrency should be managed, so that the listener does not block writing events when the profiler is accessing data.

5.1.3. Profiler

The bottleneck of the profiler will most likely be the database access speed, since it needs to access all the flow information for all users to create the model. The profiler will calculate quantiles, group samples and generate pdfs. These general-CPU operations are not easily parallelizable.

Additional memory will allow several users' flow information to be fetched at the same time. This will increase the speed, because less database queries will need to be done. On one extreme, one query per user will have to be made; on the other extreme, one query will fetch all data for all users.

5.1.4. Roler

The Roler will be CPU and memory bound. The clustering algorithm allows parallel CPU operations, if parallel implementations are available. If the entire flat data set fits in memory, the fastest and simplest implementations that use flat matrixes can be used (like the ones in R). However, if a flat dataset does not fit into memory, which is highly likely, it is preferable to use a sparse clustering algorithm. The worse option is to use a simple clustering algorithm with flat dataset larger than memory and use swapping. This will have a potential high slowdown (due to disk access) of several orders of magnitude.

5.1.5. Anomaly detector

The anomaly detector is a simple step, a simple computation of distance against the cluster centers.

5.1.6. Temporal analysis

A temporal analysis is a sequence of <Profiler,Roler,Anomaly Detector> steps for different time periods. Previous results should be cached, since they will not change. Only the last period needs to be calculated.

If sliding windows are used, the profiler can cache intermediate sample data on a minimum granularity (day) to avoid repeated database access. For example, pre-aggregated daily flow counts.

5.2. Scalability analysis

At first sight, it may seem the system will have scalability issues. The space requirements are dominated by the user profile matrix, needed by the roler. It is O(users*applications), more precisely, O(users*number_attributes*resolution*distinct_applications). In our particular case, with resolution=12, and number_attributes =3, it's O(users*36* distinct_applications).

A quick estimate shows that, for 10.000 users and 10.000 applications there will be 10K*10K*36 = 3,6G elements. Using 32-bit floats that's a total of 13,4GB! The following sections explain why this estimate is inaccurate.

5.2.1. User / distinct application ratio

It is important to realize that the number of distinct applications does not grow linearly with the number of users, even if we consider irrelevant applications. Using the complete (two months)

dataset, for every X (number of users), 10 random users were picked, and the number of distinct applications calculated. Figure 24 shows the resulting plot.

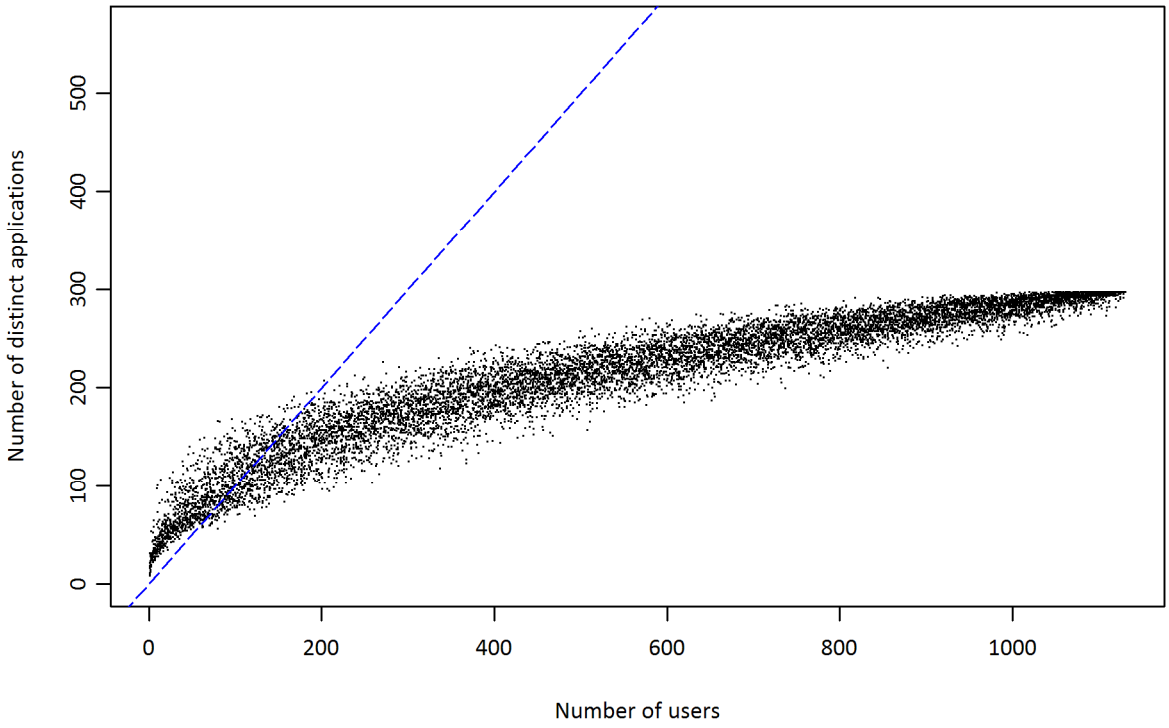

Figure 24 – User to distinct application ratio

The blue dashed line is the 1:1 ratio (y=x). It can be seen that the real ratio is much better than linear, it is either logarithmic or linear with a low slope. It seems it can be approximated with a 1:4 ratio.

5.2.2. Sparseness

Another point to consider is that not all users use every application. Let's consider the first month data. We have 841 users and 203 distinct applications. The matrix would have 841 X 203 elements = 170723 elements (<user,application> pairs). However, since not every user uses every application, only 13147 elements are not 0. This yields a ratio (real elements / maximum elements) of 7.7%.

For two months data the numbers are similar. We have 1131 users and 231 distinct applications. 18729 elements are not 0 over a possible 335907, which yields a ratio of 5.6%. Table 11 sums up these results.

Period	Users	Applications	Max. Elements	Real Elements	Ratio
One Month	841	203	170723	13147	7,7%
Two Months	1131	231	335907	18729	5,6%

Table 11 – User / application sparseness (application granularity)

5.2.3. Sparseness refinement

We can refine the previous approach, and consider that some pdf slices may have a probability of zero. If some elements of each pdf can be zero, the sparseness factor may be even lower. Using the

previous one month example, recalling that an application is, in fact, three pdfs with 12 slices each, with 841 users and 203 distinct applications, the matrix would have 841 X 203*(3*12) = 6146028 elements (pdf slices). However, only 149845 elements are not 0. This yields a ratio of 2.4%, much better than the previous 7.7%.

Similarly for two months data, the matrix would have 1131 X 231*(3*12)= 12092652 elements. Since only 210710 elements are not 0, the ratio drops from 5.6% before to 1.7%.

Period	Users	Applications	Max. Elements	Real Elements	Ratio
One Month	841	203	6146028	149845	2.4%
Two Months	1131	231	12092652	210710	1.7%

Table 12 – User / application sparseness (pdf slice granularity)

These results mean that sparse-efficient X/K-means algorithms can and should be implemented. These algorithms will have a small computational overhead for lookup operations dealing with the sparse data structures, but these are several orders of magnitude smaller than the penalty incurred by the entire flat matrix not fitting into memory – the swap disk I/O access times would severely impact performance.

5.2.4. Projection

Consider the same 10.000 user deployment scenario from the introduction of this section. This time we will use the techniques and estimates presented so far. First, the estimated number of distinct applications is 2500, using the 1:4 ratio. The matrix will be 10000 X 2500. With three fields and resolution=12, it will have 10K X 2.5K*3*12 values. Using a conservative 5% sparsiness factor, there will be 45M elements. Using 32-bit floats, the matrix will occupy around 172MB, and using 64-bit doubles 343MB. Even if we account for overhead due to the sparse structure management, this is a reasonable memory footprint, causing no problem whatsoever.

5.3. Privacy

In many areas of the world, like in Europe, the privacy of the workers is a fundamental right. Considering a process in which this system is inserted, the output of the anomaly detector and the roler will be analyzed by a human operator. These operators could be unnecessarily exposed to their co-workers private data (which applications they use, etc). The following is a suggested workflow to handle the anomalous behavior, reducing private data exposure to a minimum.

The suspected anomalies will be presented to system specialists, operators skilled in workstation administration so that they recognize the applications. No information about the user will be displayed (username, name, id, number, etc.), except for his class (department). With the application and cluster information, a correct decision can be taken whether the user should be flagged for analysis.

The system will send a notification to the flagged users' hierarchical supervisors, asking if the anomalous behavior is justifiable. If it is, exceptions (whitelisting) will be created, so that they are not flagged again. If not, more specialized tools can be activated to analyze the situation.

6 Future work

Our work's modular approach allows for independent evolutions and parallel research of each module. During the course of our work, the following topics or areas were considered interesting to analyze in future work.

Temporal Analysis. Temporal analysis concerns the evolution over time of several of the parameters. A temporal analysis of the anomaly score per user will allow making a distinction between single occurrences (spikes), and persistent behavior. It will also allow verifying if the anomaly score for a user is stable, increasing or decreasing. A temporal analysis on the cluster stability will show how the roles are evolving over time, and if they are stable or show much variation. It will be interesting to known which users changed clusters, and how they relate with the anomalies.

Web-based applications. Web-based applications are accessed using the same client application – the web browser. Because of this, all of their flows are mixed together in the web browser application. In effect, we are analyzing all their behaviors as one application. This may lead to imprecise results and application masking. To solve this, we would create virtual applications: unfold the browser application into several <browser,ip> virtual applications. It is very rare to have web services for different applications on same machine. Therefore, using IP addresses (or ranges), we could tell these applications apart. The same goes for internet traffic, since in most corporations it flows through a web proxy. Internet traffic could also have its own virtual application.

Variable-sized slices in pdf. Our current approach uses fixed-sized slices (or bins) and calculates a quantile to avoid outlier compression. This technique is simple and effective, but the higher quantile value must be manually adjusted. For example, from our analysis we suspect that the 99% quantile may still be compressing the duration attribute.

The approach used in Botminer [15] is interesting, it uses variable-sized slices. As many quantiles as the number of slices are calculated, so that each contains approximately the same number of samples. For example, in our r=12 case, 12 quantiles would be calculated, each with 8.3% of the sample space. Bin one would have samples from 0 to the value of the 8.3% quantile, bin two from the 8.3% quantile to the 16.7% quantile, and so on, until the last one, bin twelve from the 91.7% quantile upwards.

However, some questions remain, like, how would this variable-sized bins algorithms affect the pdf area properties?

Weighted k-means/x-means. Currently every application has the same weight (importance). The idea is that giving them different weights could cluster the users differently, closer to reality.

Automatic anomaly evaluation. Anomaly verification will most likely be a manual process, especially if no false positives are tolerated. As the operators accumulate experience over time, they will build a mental database of 'suspect' applications. By transferring this knowledge into the system, each application could have an 'anomalous level score'. The goal is for the system to automatically

calculate an anomalous level score for the user, a refinement of the anomaly score, speeding up the task of the operator – or maybe even completely automating the system, except for new applications.

Cluster Refinement. The clustering algorithms used do not have the ability to automatically mark samples as noise. Every sample must forcibly belong to a cluster. This means that anomalies may be skewing the cluster centers. Each time an anomaly is flagged, the clusters could be recalculated without the said anomaly (outlier). This would improve the clustering accuracy.

Multi-dimensional pdf. Our current model uses three independent pdf, one for each attribute. However, these attributes may not be (as very likely are not) independent. To capture the dependencies between them, a multi-dimensional pdf (in our case tridimensional) pdf could be used. Our current model of three independent pdf can be seen as projections of this three-dimensional pdfs into their own plane. This would inevitably bring additional challenges, for example, how to visualize the data. Do not forget that the function would be of the kind R^3 to R, requiring four dimensions to visualize. It might also have an effect on the pdf area properties.

Other clustering algorithms. By taking advantage of the modular design, other clustering algorithms and techniques could be experimented and compared with the current implementation. For example, graph-based or density-based clustering. It would be interesting to analyze the density-based clustering capability of marking samples as outliers.

PCA and K-means relation. Work by He et al. in [38] and [39] shows that PCA automatically projects to the subspace where the global solution of K-means clustering lie, and thus facilitate K-means clustering to find near-optimal solutions. Maybe PCA can be used to improve the clustering algorithm after all.

Model evolution. Finally, the model itself can be refined. Can it be simplified and still achieve comparable results? Alternatively, maybe it could be extended capture the user behavior with more detail.

7 Conclusion

The goal of this project was to use network flows tagged with user and application information to characterize the user network behavior. The knowledge of how users are utilizing which application can have an impact on the company IT infrastructure management. By knowing their real needs, resource usage can be optimized, and facilitate relocation. It may also detect anomalous behavior, a possible indicator of insider threat. In addition, the compliance with the security policy can be verified.

With those goals in mind, we proposed a model and system architecture to characterize user network behavior, group them into roles (similar groups) and detect anomalies. This characterization can be used to identify each role (and user) network requirements, and to identify deviant behavior.

The architecture consists of four components. The probes and event repository collect and store the tagged network flow information. The profiler characterizes each user behavior by creating an individualized profile. The roler groups user patterns into groups (roles) that reflect similar behavior, and creates role profiles. Finally, the anomaly detector compares the users' profiles against the roles profiles and flags anomalous behavior. The user and role profile model consists of per application flow probability distribution functions of three attributes: flow duration, flow time of day and flow traffic.

To evaluate our system and validate the model, we implemented a prototype and tested it using a real dataset. The dataset consisted of two months of recent data captured in a callcenter with over a thousand distinct users, and over two hundred distinct applications. The results confirm that the roles accurately map similar behavior. The anomaly results were also what we expected, considering the underlying populations.

Visualization methods were created to quickly visualize all the information the model contains. The User/Application spectrogram allows a bird's eye view of the user profiles and the roles differences. The Application energy plot shows a signature for each role, allows role comparison, and identify which applications are used the most. The anomaly score plot clearly shows how anomalous some behaviors are. Finally, but not less important, the three side-by-side pdfs allow to characterize an individual application.

We conclude that our work is a first step into better understanding and characterizing the user network behavior. We believe that our model is sound, and that it effectively and accurately captures the user behavior and flags anomalies. If deployed with the recommended precautions, it can be a valuable tool for any corporation. With the knowledge that it can mine from the raw data, the users network needs could be better fulfilled, the anomalous users flagged for inspection, giving an edge in agility for any company that used it.

8 References

[1] J. Saltzer and M. Schroeder, “The protection of information in computer systems,” *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[2] T. Mendo, “Document flow tracking within corporate networks,” Master’s thesis, Carnegie Mellon University / Faculdade de Ciências da Universidade de Lisboa, November 2009.

[3] D. Luckham, *The power of events: an introduction to complex event processing in distributed enterprise systems*. Springer, 2002.

[4] J. A. Alegria, T. Carvalho, and R. Ramalho, “Uma experiência open source para "tomar o pulso" e "ter pulso" sobre a função sistemas e tecnologias de informação,” *5th CAPSI*, Lisboa, 2004.

[5] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[6] H. Hotelling, “Analysis of a complex of statistical variables into principal components,” *Journal of educational psychology*, vol. 24, no. 6, pp. 417–441, 1933.

[7] G. Golub and W. Kahan, “Calculating the singular values and pseudo-inverse of a matrix,” *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, vol. 2, no. 2, pp. 205–224, 1965.

[8] J. MacQueen, “EnglishSome methods for classification and analysis of multivariate observations,” Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297, 1967.

[9] D. Pelleg and A. W. Moore, “X-means: Extending k-means with efficient estimation of the number of clusters,” in *ICML ’00: Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 727–734.

[10] M. Ester, H. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proc. KDD*, vol. 96, 1996, pp. 226–231.

[11] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[12] A. Patcha and J. Park, “An overview of anomaly detection techniques: Existing solutions and latest technological trends,” *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[13] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski, “Clustering approaches for anomaly based intrusion detection,” *Proceedings of Intelligent Engineering Systems through Artificial Neural Networks*, pp. 579–584, 2002.

[14] E. Parzen, “On estimation of a probability density function and mode,” *The annals of mathematical statistics*, pp. 1065–1076, 1962.

[15] G. Gu, R. Perdisci, J. Zhang, and W. Lee, “Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection,” in *Proceedings of the 17th annual USENIX Security Symposium*, August 2008, pp. 139–154.

[16] J. Erman, M. Arlitt, and A. Mahanti, “Traffic classification using clustering algorithms,” in *MineNet ’06: Proceedings of the 2006 SIGCOMM workshop on Mining network data*. New York, NY, USA: ACM, 2006, pp. 281–286.

[17] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, “Blinc: multilevel traffic classification in the dark,” in *SIGCOMM ’05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 229–240.

[18] M. Roughan, S. Sen, O. Spatscheck, and N. Duffield, “Class-of-service mapping for qos: a statistical signature-based approach to ip traffic classification,” in *IMC ’04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004, pp. 135–148.

[19] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “Acas: automated construction of application signatures,” in *MineNet ’05: Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. New York, NY, USA: ACM, 2005, pp. 197–202.

[20] G. Tan, M. Poletto, J. Guttag, and F. Kaashoek, “Role classification of hosts within enterprise networks based on connection patterns,” in *Proceedings of USENIX Annual Technical Conference*, 2003, pp. 15–28.

[21] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher, “Web page categorization and feature selection using association rule and principal component clustering,” in *In 7th Workshop on Information Technologies and Systems*, 1997.

[22] E. Han, G. Karypis, V. Kumar, and B. Mobasher, “Clustering based on association rule hypergraphs,” *UMSI research report/University of Minnesota (Minneapolis, Mn). Supercomputer institute*, vol. 98, p. 15, 1998.

[23] E.-H. Han, G. Karypis, V. Kumar, and B. Mobasher, “Hypergraph based clustering in high-dimensional data sets: A summary of results,” *IEEE Bulletin of Technical Committee on Data Engineering*, vol. 21, no. 1, pp. 105–140, 1998.

[24] E. Han, G. Karypis, and V. Kumar, “Min-apriori: An algorithm for finding association rules in data with continuous attributes,” Department of Computer Science and Engineering, University of Minnesota, Tech. Rep., 1997.

[25] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, “Discovery and evaluation of aggregate usage profiles for web personalization,” *Data Mining and Knowledge Discovery*, vol. 6, no. 1, pp. 61–82, 2002.

[26] T. Henderson and S. Bhatti, “Modelling user behaviour in networked games,” in *MULTIMEDIA ’01: Proceedings of the ninth ACM international conference on Multimedia*. New York, NY, USA: ACM, 2001, pp. 212–220.

[27] G. Box, G. Jenkins, and G. Reinsel, *Time series analysis: forecasting and control*. Holden-day San Francisco, 1976.

[28] J. S. Park and J. Giordano, "Role-based profile analysis for scalable and accurate insider-anomaly detection," in *Proc. 25th IEEE International Performance, Computing, and Communications Conference IPCCC 2006*, 10–12 April 2006, pp. 7pp.–470.

[29] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural analysis of network traffic flows," in *SIGMETRICS '04/Performance '04: Proceedings of the joint international conference on Measurement and modeling of computer systems*. New York, NY, USA: ACM, 2004, pp. 61–72.

[30] E. Hoke, J. Sun, J. D. Strunk, G. R. Ganger, and C. Faloutsos, "Intemon: continuous mining of sensor data in large-scale self-infrastructures," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 3, pp. 38–44, 2006.

[31] S. Papadimitriou, J. Sun, and C. Faloutsos, "Streaming pattern discovery in multiple time-series," in *Proceedings of the 31st international conference on Very large data bases*. VLDB Endowment, 2005, pp. 697–708.

[32] E. Manavoglu, D. Pavlov, and C. Giles, "Probabilistic user behavior models," in *Third IEEE International Conference on Data Mining, 2003. ICDM 2003*, 2003, pp. 203–210.

[33] G. Karypis and E. Han, "Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization," Department of Computer Science and Engineering, University of Minnesota, Tech. Rep. 00-16, March 2000.

[34] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, vol. 1215, 1994, p. 487499.

[35] C. Ware, *Information visualization: perception for design*. Morgan Kaufmann, 2004.

[36] The r project for statistical computing. http://www.r-project.org/.

[37] Rjava. http://www.rforge.net/rJava/.

[38] H. Zha, X. He, C. Ding, and M. Gu, "Spectral relaxation for k-means clustering," Technical Report TR-2001-XX, Pennsylvania State University, University Park, PA, Tech. Rep., 2001.

[39] C. Ding and X. He, "K-means clustering via principal component analysis," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM, 2004, p. 29.