ABSTRACT

Title of dissertation: USER BEHAVIORAL MODELING OF WEB-BASED SYSTEMS FOR CONTINUOUS USER AUTHENTICATION

Leslie C. Milton, Doctor of Philosophy, 2015

Dissertation directed by: Professor Atif M. Memon
Department of Computer Science
University of Maryland, College Park

Authentication plays an important role in how we interact with computers, mobile devices, the web, etc. The idea of authentication is to uniquely identify a user before granting access to system privileges. For example, in recent years more corporate information and applications have been accessible via the Internet and Intranet. Many employees are working from remote locations and need access to secure corporate files. During this time, it is possible for malicious or unauthorized users to gain access to the system. For this reason, it is logical to have some mechanism in place to detect whether the logged-in user is the same user in control of the user's session. Therefore, highly secure authentication methods must be used.

We posit that each of us is unique in our use of computer systems. It is this uniqueness that is leveraged to "continuously authenticate users" while they use web software. To monitor user behavior, n-gram models are used to capture user interactions with web-based software. This statistical language model essentially captures sequences and sub-sequences of user actions, their orderings, and temporal

relationships that make them unique by providing a model of how each user typically behaves. Users are then continuously monitored during software operations. Large deviations from "normal behavior" can possibly indicate malicious or unintended behavior. This approach is implemented in a system called *Intruder Detector* (**ID**) that models user actions as embodied in web logs generated in response to a user's actions. User identification through web logs is cost-effective and non-intrusive. We perform experiments on a large fielded system with web logs of approximately 4000 users. For these experiments, we use two classification techniques; binary and multi-class classification.

We evaluate model-specific differences of user behavior based on coarse-grain (i.e., role) and fine-grain (i.e., individual) analysis. A specific set of metrics are used to provide valuable insight into how each model performs. *Intruder Detector* achieves accurate results when identifying legitimate users and user types. This tool is also able to detect outliers in role-based user behavior with optimal performance. In addition to web applications, this continuous monitoring technique can be used with other user-based systems such as mobile devices and the analysis of network traffic.

USER BEHAVIORAL MODELING OF WEB-BASED SYSTEMS FOR CONTINUOUS USER AUTHENTICATION

by

Leslie C. Milton

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Atif Memon, Chair
Professor Rance Cleaveland
Professor Udaya Shankar
Professor Alan Sussman
Professor Carol Espy-Wilson

Dedication

"For I know the plans I have for you, declares the Lord, plans to prosper you and not to harm you, plans to give you hope and a future." Jeremiah 29:11 (NLT)

To God; my incredible fiancé and best friend, Raymond Leonard, Jr.; my amazing parents, Linda and Eddie Milton, Jr., and all of my friends and family. Thank you for taking this journey with me. Your love, support and encouragement helped me complete this chapter of my life.

Acknowledgments

With God, all things are possible. I would like to thank God for blessing me with the ability to complete this chapter of my life. I would like to thank my amazing parents, Linda and Eddie Milton Jr., whose love consumed, motivated, and comforted me. Thank you for making the goal of obtaining a Ph.D. achievable.

I would like to thank my wonderful advisor, Dr. Atif Memon. I could not have asked for a more patient, humble, and enthusiastic advisor. Your devotion and guidance has helped me become a better researcher. I would like to thank my committee members; Dr. Ashok Agrawala, Dr. Rance Cleaveland, Dr. Udaya Shankar, Dr. Alan Sussman, and Dr. Carol Espy-Wilson for their insightful comments and feedback.

I am extremely grateful to the U.S. Army Engineer Research and Development Center (ERDC) for funding my doctoral studies. I appreciate ERDC's leadership team and my wonderful co-workers for being a great support system, including, but not limited to: Dr. Reed Mosher, Patti Duett, Ken Pathak, Renee Mullinax, Dr. Louis Turcotte, Dr. Kevin Abraham, Dr. Elaine Hulitt, Dr. Gerald Morris, John West, Marvin Brown, Jason Powers, Jimmie Baker, Patrick Gladney, Krisa Rowland, Cathy Ballard, Cheryl Sims, Shari Thomas, Chris Lunderman, Marcus Randle, and William Young.

I appreciate the Faculty, Staff, fellow graduate students, and friends that I have worked with while attending the University of Maryland, including but not limited to: Dr. Howard Elman, Dr. Nick Roussopoulos, Dr. Ben Shneiderman, Dr.

Amol Deshpande, Dr. Hal Daume III, Dr. Hanan Samet, Dr. T. Kanti Srikantaiah, Dr. Gleenesha Williams, Vera Rhoads, Fatima Bangura, Jennifer Story, Bryan Robbins, Dzung (Bryan) Ta, Zebao Gao, Emily Kowalczyk, Ethar Elaska, Ishan Banerjee, Bao Nguyen, Richard Johnson, Travis Finkenauer, Ray Chen, Deonna Hodges, Greg Sanders, Wikum Dinalankara, Emre Sefer, Austin Meyers, Christine Lu, Tandeep Sidhu, and Sonia Ng.

I would like to thank my wonderful mentors for sharing their personal, academic, and professional expertise, including but not limited to: Dr. Deborah Dent, Dr. Keith Hudson, Dr. Natarajan Meghanathan, Dr. Charles Bland, and Dr. Cyntrica Eaton.

I am thankful to my family and friends for playing an important role in the completion of this work. Special thanks to Patrick Milton, Lamonte Allen, Patrice Milton, Maurtice Milton, Anthony Sims, Jr., Larkale Stokes, Marshall Thompson, Jr., Dr. Jamese Sims, and Sheena Caston.

I would like to thank my fiancé, Raymond Leonard, Jr., for taking this journey with me. Your unconditional love, support, and encouragement motivated me to push forward.

Finally, I would like to express my sincerest gratitude to *everyone* who has impacted my life in a positive way. There are not enough pages to exhaustively list the abundant number of incredible supporters that I have encountered in my lifetime. To everyone that supported my endeavors, please know that I appreciate each of you.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Many web-based applications rely on authentication methods that are reliable, convenient and secure. Username and password have been universally accepted by most applications to be the only form of authentication. Some systems require the use of long passwords that need to be changed frequently. They can be difficult to remember, create, and manage [1]. In addition to long passwords, passwords that are too short or lack complexity also pose a significant risk. In a study of over 3.3 million leaked passwords from North America and Western Europe, SplashData records "123456" and "password" as the top two passwords chosen by users [2].

Conventional authentication methods do not ask the user to verify their identity during their active log-in session, leaving the computer system vulnerable to malicious or unintended use while the user is logged-in [3]. To improve the authentication process for web-based applications, there must be a method to continuously verify the identity of a user. *Continuous User Authentication* (CUA) has been proven to solve this limitation. CUA techniques monitor, verify, and authenticate users during their entire session. CUA generates user profiles and compares them to the user's stored profile. If user activity deviates from its normal pattern of usage, the system generates an alarm. CUA systems have user profiles that are customized

for every application. This makes it difficult for attackers to know which actions will be detected as intrusive [4].

Several studies have used biometrics to continuously authenticate users by the use of cognitive fingerprints, eye scans, color of user's clothing, and face tracking [3] [5] [6]. However, many of these techniques require additional hardware and cost to operate efficiently. Behavioral modeling addresses these limitations by monitoring how users interact with the system. Evaluating mouse movement, how users search for and select information, and the habitual typing rhythm of users are measures used to continuously observe a user's behavior [7][8]. Although these approaches do not require special hardware, most of them require the installation of specialized monitoring software.

This research addresses challenges that occur when modeling the behavior of users that interact with web-based organizational information system applications. These applications run inside a Web browser-based front-end and are accessible via hypertext transfer protocol (HTTP). It also includes middleware to implement business logic and a back-end database. We categorize information system as organized systems for the collection, organization, communication and storage of information [9]. We develop a new tool, *Intruder Detector* (**ID**), to model unique behavioral footprints for each user. Patterns of use for a specific user or group of users is captured in this footprint and leveraged to "continuously" authenticate the user. **ID** performs behavioral analysis for each user and builds a context of each user's behavior, based on a statistical language model, to verify the user's identity. No additional hardware is required to deploy this tool. Furthermore, we seek to provide a cost-effective and

non-intrusive solution for web-based user authentication. Our preliminary work for this research received the Best Paper Award at the *Eighth International Conference on Emerging Security Information, Systems, and Technologies* [10].

We provide the following contributions from this research:

- We develop a novel keyword abstraction technique to pre-process large volumes of web logs by eliminating incomplete, noisy and inconsistent data.
- We use statistical language models to capture the behavior of users while they interact with organizational web-based applications.
- We develop a continuous user authentication framework with the ability to categorize user sessions into a predefined set of roles or finer-grained user profiles. This framework is also used to identify outliers in role-based user behavior.
- We introduce a set of evaluation metrics to test the feasibility of our approach.

1.1 Continuous User Authentication Scenarios

This research explores how modeling user behavior affects system security and usability when using data that is already available (e.g., web server logs). The following two scenarios show how CUA may be used in typical settings.

Scenario 1: *Alice* uses her laptop to *telework* from a local coffee shop. She uses her web browser to login to her corporate site and perform her daily tasks. As she steps away from her laptop to get a refill of coffee, an unauthorized person

accesses her laptop to perform actions on the corporate site, maliciously merging and changing sensitive records. Unbeknownst to the "intruder," the web-site is equipped with our CUA system, called *Intruder Detector*, that automatically detects deviations from normal behavior and subsequently locks her computer. **ID** works as follows: It monitors all user actions and builds, for each user, an n-gram model[1], a mathematical representation of how the user typically interacts with the web application. **ID** determines, in real time, whether the user is deviating from expected behavior, signaling the possible presence of an intruder. Because this instance of **ID** is based solely on analysis of web logs, it is extremely fast and it requires no special hardware or changes to the web application.

Scenario 2: *Bob's* office computer is used to perform illegal transactions via the company's web application. Because the computer is located in a "secure" area, the forensics team concludes that the malicious transactions must have been done by a co-worker. They retrieve the system's web access logs and filter out those that originated from *Bob's* computer, thereby obtaining all the web requests between *Bob's* computer and the web server. They then use these logs and the models stored in **ID**, which represent exactly how each user typically behaves when using the web application, to pinpoint the co-worker who most closely resembles the pattern of unauthorized accesses. In this scenario, there is some level of uncertainty because the malicious sequences by the intruder may or may not match their stored user model. At this point, **ID** is simply one tool in the investigative process and

[1] An n-gram representation models sequences using the statistical properties of n-grams (contiguous sequences of n items/actions).

additional forensic data is needed to pinpoint the person performing the malicious transactions.

1.2 Thesis Statement

This work aims to obtain a unique behavioral footprint indicating patterns of use for specific users or group of users of a web-based organizational information system application using information that is naturally generated by the system (e.g., web logs). Models are then constructed for any targeted group of sessions based on n-grams. By leveraging this footprint, users are "continuously" authenticated. This leads to the following thesis statement:

Sequences and subsequences of user action that is obtained from web log files can be captured within statistical language models to continuously authenticate users of web-based organizational information system applications.

1.3 Approach

To prove the above thesis statement, we analyze how log data from web-based applications can be used to predict user behavior. Web server logs capture all requests made to the server. For many systems, web server log files are not fully utilized. These logs, also called access logs, include historical information about the activities performed by users. There are several ways web server logs can be used to present valuable information to a system administrator. For example, information extracted from log files have been used by companies to provide better service to

customers and improve the quality of their website.

In addition to improving user interaction, web logs can be used to trace patterns of behavior. The patterns can then be used with statistical language models, such as n-grams, to predict user behavior. n-gram models have performed surprisingly well in the domain of natural language processing (NLP), where researchers have found that a history of only one to two events is necessary to obtain optimal predictive capabilities [11]. An n-gram model captures all sequences of a fixed length, N, from previously observed user input, which allows prediction and evaluation of future behavior based on frequencies. In the realm of NLP, these sequences are words generated for sentences. Web-based user actions also have a grammar-like sequential structure. If it is assumed that the event sequences carried out by users of a software system are analogous to natural language, we would expect to find the same predictive power in n-gram models of software event sequences as well.

There are several NLP models that could be used for this research. For example, Hidden Markov Models (HMMs), are much more elaborate models with the ability to track independent probability distributions, including those of hidden variables. We explore several NLP models and determine that n-gram language models provide predictive power when modeling user behavior based on web logs. Understanding the behavior and utility of n-grams to build user profiles is a reasonable prerequisite for using more advanced machine learning methods.

This work addresses the following challenges and makes the following corresponding contributions to the realm of web-based user profile analysis:

Challenge 1: Several models exist for the analysis of user behavior.

Contribution 1: In this work, we explore several statistical language models. We successfully identify a suitable model for web-based usage behavior.

Challenge 2: There is no straightforward method to identify the behavior of users that interact with web-based applications.

Contribution 2: We develop a novel keyword abstraction process to identify user activity for each system user.

Challenge 3: It is complex to monitor user behavior without the use of specialized hardware.

Contribution 3: We develop new techniques to monitor user behavior of web-based applications using information that is already available to a system administrator.

Challenge 4: There is a trade off in performance when developing a CUA system.

Contribution 4: We test the performance of our implementation using a large set of evaluation metrics to access the feasibility of our approach.

Our final contribution is an empirical analysis of the CUA technique. This method is applied to a government fielded training support website. Over a period of three years, approximately 4000 users, with one of four access levels, were captured in the dataset.

1.4 Research Scope

There are several types of web-based information systems. In this research, to provide focus, we only consider web-based organizational information system applications described in Figure 1.1[2]. These systems include executive, senior, middle, and worker-level access usage. To access these applications, employees must use the organization's network with an option to connect via virtual private network (VPN). Modeling common websites without an organizational focus, such as www.amazon.com, are beyond the focus of this research.

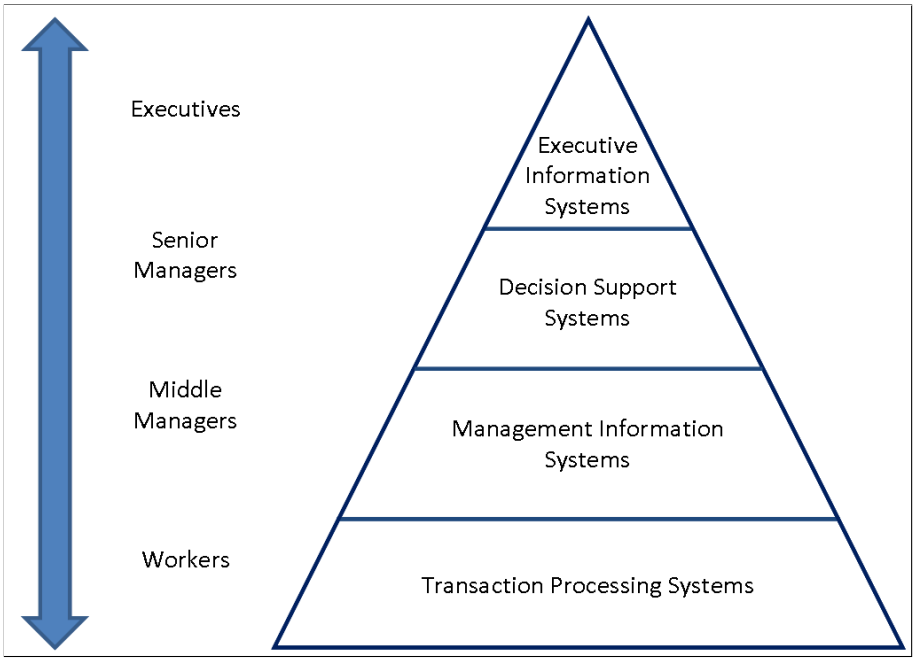


Figure 1.1: Organizational-level Information Management Systems

[2]https://en.wikipedia.org/wiki/Information_system

1.5 Broader Impact

We believe the work presented in this research will provide an increased level of security when combined with existing authentication techniques (e.g., passwords, biometrics) to support many demands of today's computing environment. While conducting this research, we have witnessed how important it is to evaluate user behavior and provide a secure environment for our users to conduct their day-to-day business. After reviewing the web logs for this study, we identified several IP addresses that were not indicative of the interaction of approved users. This prompted the information assurance team to launch an investigation and put measures in place to stop this activity. In this case, **ID** could have been utilized to identify and stop interactions that do not closely resemble stored usage profiles.

Besides its importance to the realm of security, modeling user behavior, in general, is an important process for the customization and adaptation of user-specific needs[3]. Dynamic user modeling gives a current picture of how the user interacts with the software. As a user's interest and behavior changes over time, adaptive learning can be applied to increase the predictive power of the models. This is needed for CUA to be effective in this domain. We believe n-grams are highly effective models that can be used to capture user interactions as sequences by aiding in the identification of patterns that deviate from normal behavior.

[3] `http://en.wikipedia.org/wiki/User_modeling`

1.6 Dissertation Outline

The remainder of this dissertation is organized as follows. In Chapter 2, we present background and related work for statistical language models and continuous user authentication. Chapter 3 highlights the contributions of our work. Specifically, we show how *Intruder Detector* is used in typical settings and the valued-added for this approach. In Chapter 4, we provide a detailed empirical evaluation and present results for our CUA implementation. Finally, Chapter 5 contains an overview of our work and areas for future research.

Chapter 2: Background and Related Work

In this chapter, we present classical forms of authentication and describe the need for non-intrusive, continuous authentication methods. We define and review existing approaches in the CUA domain. We also focus on the use of Statistical Language Models (SLM) to capture sequential behavior of users that interact with web-based organizational information systems. A survey of these techniques is outlined in the following sections to add context to this research.

2.1 Classical Authentication Methods

Various authentication methods exist. Passwords, smart cards, digital certificates, kerberos, and biometrics are among the many authentication methods currently employed. There are three classical forms of authentication: (1) something the user knows; e.g. password, pin, (2) something the user has; e.g. smart card, Yubikey [12], and (3) something the user is; e.g. iris scan, fingerprint.

These authentication mechanisms are useful but have well known limitations. A single point-of-failure exists if a user's password is compromised and later accessed by a malicious user. In addition, the patterns used when creating a password warrants some concern because keystroke analysis indicates that users often resort

to special patterns [13]. If a device is found unlocked or hijacked during a user session, the system becomes compromised. Many biometric techniques also require additional hardware to collect data. Biometrics are not secret [14]. Intruders can observe a user's features and attempt to manipulate the system. Many companies and organizations use Multi-Factor Authentication (MFA) to mitigate this risk. MFA uses two or more classical forms of authentication to gain access to the system. If one form of authentication is cracked, guessed, or otherwise stolen, an attacker's access is still prohibited.

The use of MFA has become more complex over time. Many banks offer special apps and web logins but do not require the same MFA method when a customer accesses their account from an ATM. User productivity can also decrease because of the time it takes to use multiple authentication methods. For example, if a company requires two-step authentication for 1000 employees using a password and smart card; there will be 8000 login transactions if each user logs in 4 times per day (8 entries). This also adds overhead to the system and multiple points-of-failure. It is difficult to identify an intruder when using this form of authentication.

Various research studies have explored the use of authentication. Kaminsky *et al.* address challenges for user authentication in a global file system [15]. This approach uses an authentication server to identify users based on local information. Researchers of cloud computing security methods have developed implicit authentication to identify a user's past behavior to authenticate mobile devices [16]. These two studies have one major limitation worth noting. They lack the ability to continuously monitor a user's behavior as they interact with the system.

2.2 Defining Continuous User Authentication

Cybersecurity has become a key concern for many organizations and companies. For example, the Office of Personnel Management (OPM) informed millions of government and military employees that their personal information may be compromised [17]. For many systems, the first line of defense is authentication. Google and DARPA agree that elaborate password rules must be abandon and the use of strong authentication should be used to avoid impersonations [18] [19].

Authentication techniques are usually performed at the beginning of a user session. This form of one-time validation is ineffective in preventing malicious and unintended use. For web applications, a password and username combination for authentication has historically been used in the context of a person's initial encounter with a computer system. Over twenty years ago, when e-commerce and secure web was first introduced, passwords were mainly a stopgap measure. It was expected that something better would replace it soon. As applications and devices evolve, this means of authentication is becoming insufficient. CUA fills this gap by transparently monitoring user activity in an effort to identify deviations from normal workflow patterns. These patterns are stored usage profiles of each user of the system. Figure 2.1 provides a simple description of how a system equipped with CUA will work. The user is asked for traditional authentication credentials (i.e., username/password) to enter the application. After successful authentication, the user begins interacting with the system. The user is asked to re-authenticated if current interactions deviate from stored usage profiles. CUA adds an additional form

of end-user authentication; something you do (e.g., typical patterns of behavior).

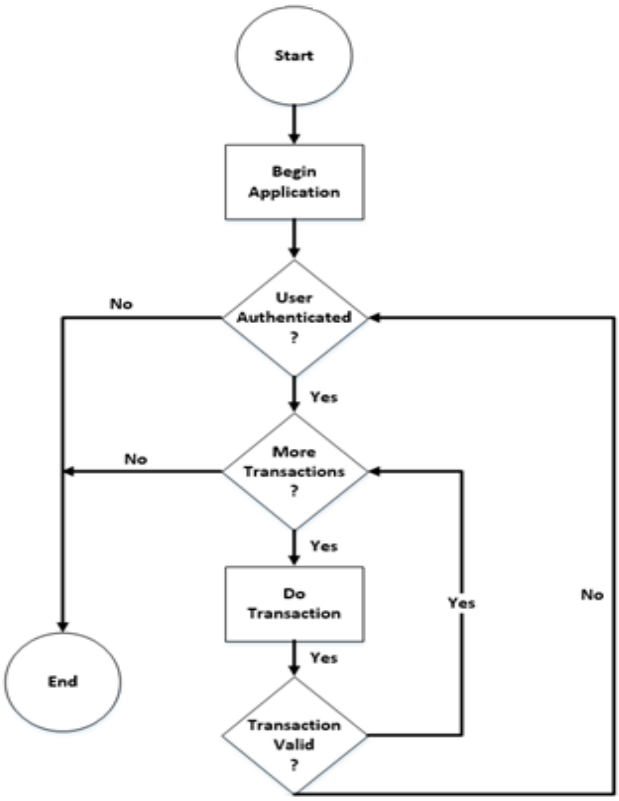


Figure 2.1: Flowchart for CUA

A robust CUA system has the following basic characteristics [20]:

- **Continual**: Re-authentication should be performed periodically to check if the current user is the logged-in user.
- **Non-intrusive**: Intrusive authentication hinders usability and provides a negative experience for the user. Therefore, the system must provide a seamless, non-intrusive user-friendly environment.
- **Behavioral**: The system must extract behavioral attributes from normal user

operations. These attributes should be cost-effective and have unique usage profiles for each user.

2.3 Existing Continuous User Authentication Techniques

The realm of CUA has been extensively evaluated with the use of biometrics. One study uses cognitive fingerprints to measure computational behavior by means of computational linguistics and structural semantic analysis [19]. This study uses a combination of metrics that include eye scans and keystrokes to evaluate how the user searches for and selects information. In addition, a number of CUA research studies use one or more hard and soft biometric traits to continuously authenticate a user. Niinuma *et al.* propose a CUA framework to automatically register the color of a user's clothing and their face as soft biometric traits [3][5]. Results from this study show that the system is able to successfully authenticate the user with high tolerance to the user's posture. Limitations to these studies exist because of the additional hardware that is needed to implement this technique which can become costly if an entire organization uses this feature to authenticate users.

Monrose *et al.* propose an authentication method that uniquely identifies users based on the analysis of keystrokes [8]. Keystroke dynamics focuses on how you type versus what you type. The habitual typing rhythm of a user is a function of the user and their environment. Therefore, limitations to this approach occur when the user is faced with environmental factors that affect their typing pattern. Altinok *et al.* propose a continuous biometric authentication system that provides an estimate of

authentication certainty at any given time, even in the absence of any biometric data [21]. In this case, the authentication uncertainty increases over time which leads to a decrease in system usability. In a similar study, Kang *et al.* introduce temporal integration of biometrics and behavioral features to continuously authenticate users [22].

A face tracking system that uses color and edge information is used to compute behavioral features. Shen *et al.* use mouse dynamics when implementing continuous user authentication [7]. This technique is used to observe behavioral features in mouse operations to detect malicious users. However, there are some existing limitations with this emerging approach. Behavioral variability occurs because of human or environmental factors. For example, if the user switches software environments or experiences biological or emotional change, the user's behavior will be modified significantly. Such changes could possibly identify the user as an impostor.

Xie *et al.* use a notable approach to identify legitimate users early when using online services by implementing a vouching process without the use of biometrics [23]. They introduce a system, Souche, to monitor vouching via social communities (i.e., Twitter, Email). Souche is effective in identifying 85% of legitimate users and denying admission of malicious users. Our research seeks to solve similar issues without the presence of social communities.

In recent years, mobile devices have been used to learn user behavior. Researchers introduced SenSec as a mobile framework to collect sensory data to construct a gesture model of how a user interacts with a mobile device [24]. Similar to our work, n-grams are used to capture user patterns. The SenSec system achieves

over 70% accuracy in user classification and authentication tasks. In addition, Saevanee *et al.* use multi-model biometric techniques with mobile devices using linguistic profiling, keystroke dynamics and behavioral profiling for user authentication [25]. Results from this study show a 91% reduction rate in the number of intrusive authentication requests.

This body-of-work extends beyond the aforementioned research studies in the following ways:

1. Instead of using traditional biometric traits, we explore the possibility of using log information that is naturally generated by web applications to improve usability through non-intrusive, transparent authentication.

2. This approach, integrated into a tool, uses a novel and simple n-gram language model to capture user behavior.

3. Experiments are based on data from actual users of a fielded Department of Defense (DOD) system who are completing day-to-day tasks.

2.4 Statistical Language Modeling (SLM)

In this research, we use SLM to classify users. This modeling technique has been successfully implemented for a variety of language-based technologies. Document classification [26], information retrieval [27], machine translation [28], and speech recognition [29] all rely on this modeling technique. Many of these models decompose

the probability of a sentence into a product of conditional probabilities [30].

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2...P(w_n|w_1w_2...w_{n-1}) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)...P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^{n} P(w_k|w_1^{k-1}) \end{aligned} \tag{2.1}$$

where w_k is the kth element in the sequence, and w_1^{k-1} $\{w_1, w_2, w_3, ..., w_n, ...\}$ is the *history, h*. Table 2.1 summarizes the SLMs explored in this research. The following subsections provide more detail on each technique.

Model	**Technique**	**Section**
Statistical Language Models	Neural Networks	2.4.1
	Maximum Entropy	2.4.2
	Probabilistic Context-free Grammars	2.4.3
	Decision Trees	2.4.4
	n-grams	2.4.5
	Hidden Markov Models	2.4.6

Table 2.1: Statistical Language Models.

2.4.1 Neural Networks

Neural networks, also referred to as artificial neural networks, for machine learning and cognitive science, was inspired by biological neural networks. Neural network models have demonstrated success in pattern recognition [31], financial modeling [32], biomedicine [33], etc. This model uses interconnected *neurons* for communication. Each connection has adaptive weights that are tuned by a learning algorithm (supervised and unsupervised). Continuous-valued features are used to automatically learn as a function of the history [34]. Artificial neurons perform the following

task:

1. Receive signals from other neurons in the network.
2. Multiply each signal by the corresponding connection strength (e.g., adaptive weight).
3. Take the sum of the weighted signals and send them to an activation function.
4. Send output to other neurons in the network.

Topology and operational mode of neural networks vary in literature. However, the most common configuration is to employ the use of an *input layer, hidden layer* and *output layer* as shown in Figure 2.2. The number of input nodes at the input layer is determined by feature values or independent variables. The output nodes are dependent on the number of classes or values to predict. To date, there is no optimal method to determine hidden nodes. If a network does not have enough hidden nodes, the input and output mappings will not be learned well. On the other hand, if there are too may hidden nodes, the network will generalize poorly on unseen elements. When neural networks operate using probability, true incremental learning is achieved. New training data can be added without retraining the entire network model [35]. A novel neural network language model is described using the following equation [36]:

$$P(w|h) = \frac{e^{\sum_{i=1}^{N} \lambda_i f_i(s,w)}}{\sum_w e^{\sum_{i=1}^{N} \lambda_i f_i(s,w)}} \tag{2.2}$$

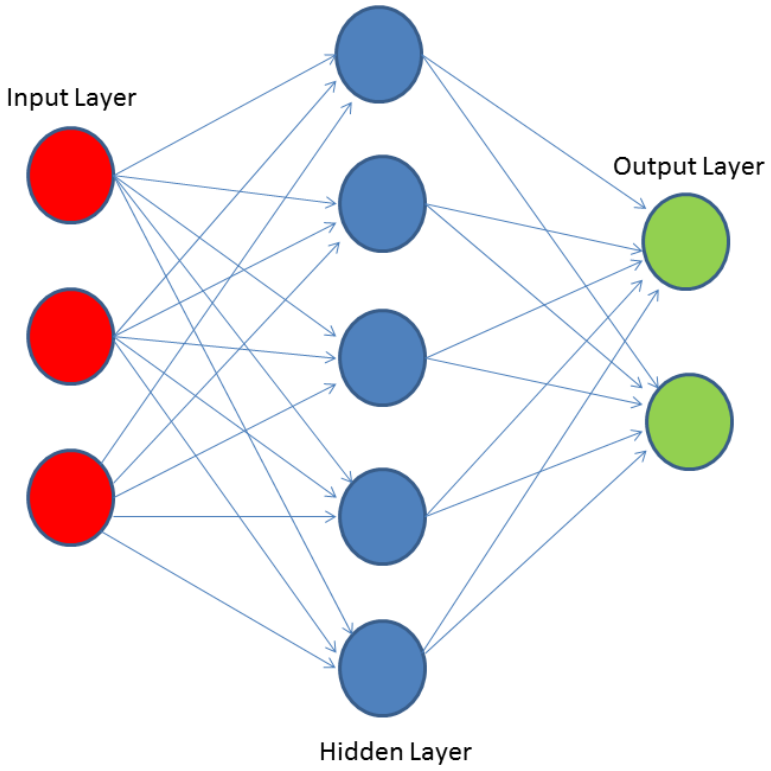


Figure 2.2: Artificial Neural Network

where e is the entropy value, f is a set of features, λ is a set of weights, h is the history and s represents the state of the hidden layer. n-grams, N, are used as the learning algorithm for equation 2.2.

Jagadeesan *et al.* use a feed forward neural network with a backward propagation method to continuously authenticate users. This implementation is based on keyboard and mouse attributes. They use the `k-nearest neighbor algorithm` for classification. Experiments with application-based user re-authentication performed at 96.4% accuracy and 3.6% false alarms. For application-independent user re-authentication, the system performed at 82%. All experiments were conducted for relatively small data sets (i.e., 5 users).

2.4.2 Maximum Entropy

Entropy represents the lack-of-order or predictability. The *Principle of Maximum Entropy* is the correct distribution of $P(a, b)$ which maximizes *uncertainty*, (i.e., entropy), based on constraints. By taking this approach, bias and assumptions are eliminated. For example, if we examine the conditional probability distribution for $P(y|x)$, then the conditional entropy is as follows:

$$H(Y|X) = -\sum_{x,y} P(x, y) log \frac{P(x, y)}{P(y)} \tag{2.3}$$

$H(Y|X)$ is the entropy of random variable Y, given the value of another random variable, X, is known [37]. When using the *Principle of Maximum Entropy*, information from many sources can be combined into one language model. Such sources can originate from n-grams with history information or local information. Rosenfeld *et al.* use maximum entropy to create a single, combined model which captures information from various knowledge sources [38]. Each knowledge source represents a particular constraint. After constraints have been identified, a set of functions are created and the function with the highest entropy (i.e., uncertainty) is chosen. This adaptive approach to maximum entropy language modeling shows approximately 39% perplexity[1], when trained on the Wall Street Journal corpus.

[1]Perplexity measures how well a statistical language model predicts a sample set of data.

2.4.3 Probabilistic Context-free Grammars

Context-free grammars consists of terminals ($w^1, w^2, ..., w^V$), non-terminals (N^1, N^2, $..., N^n$), a start symbol (N^1), and rules. Terminal symbols represent context that appear in the strings generated by the grammar. Non-terminal symbols are placeholders for patterns of terminal symbols that can be generated by non-terminals. A start symbol must be used as a special non-terminal to appear during the initial string generation. Rules are used to replace non-terminals in a string with other terminals/non-terminals.

$$
\begin{aligned}
&\langle Start \rangle \Longrightarrow X = \langle expression \rangle \\
&\langle expression \rangle \Longrightarrow number \\
&\langle expression \rangle \Longrightarrow (\langle expression \rangle) \\
&\langle expression \rangle \Longrightarrow \langle expression \rangle + \langle expression \rangle \\
&\langle expression \rangle \Longrightarrow \langle expression \rangle - \langle expression \rangle \\
&\langle expression \rangle \Longrightarrow \langle expression \rangle * \langle expression \rangle \\
&\langle expression \rangle \Longrightarrow \langle expression \rangle / \langle expression \rangle
\end{aligned}
\tag{2.4}
$$

Expressions in 2.4 is an example of a context-free grammar. This grammar is used to form a mathematical expression with five terminals as operators (+, -, *, /) and numbers. $\langle expression \rangle$ is the start symbol and the only non-terminal for this

grammar. Suppose we want to find the correct grammar to generate $X = 45 + 98 * 4$ as a mathematical expression. The context-free string generation in Figure 2.3 can be used.

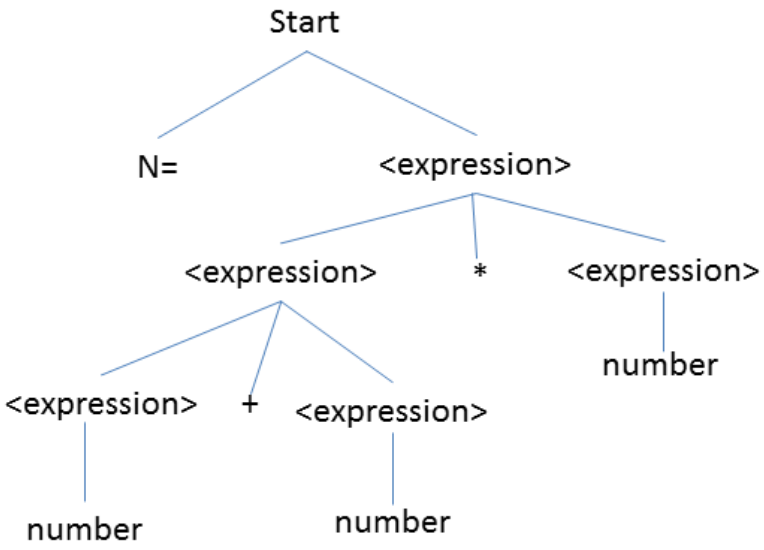


Figure 2.3: Context-free String Generation

Grammar	**Probability**
$\langle Start \rangle \Longrightarrow X = \langle expression \rangle$	1.0
$\langle expression \rangle \Longrightarrow number$	.2
$\langle expression \rangle \Longrightarrow (\langle expression \rangle)$	.11
$\langle expression \rangle \Longrightarrow \langle expression \rangle + \langle expression \rangle$	.30
$\langle expression \rangle \Longrightarrow \langle expression \rangle - \langle expression \rangle$	.15
$\langle expression \rangle \Longrightarrow \langle expression \rangle * \langle expression \rangle$	.18
$\langle expression \rangle \Longrightarrow \langle expression \rangle / \langle expression \rangle$	.06

(2.5)

Probabilistic Context-free Grammars (PCFG) assign probability estimates to each rule such that the sum of the probabilities for all rules expanding the same non-terminal is equal to one. For example, the grammar in 2.5 includes probabilities that should be used to generate an equation. If more than one trace through the grammar exists, the grammar with the highest probability is chosen. PCFGs can be learned from positive data examples alone but grammar induction is very difficult. This form of language modeling is robust and in some cases provides better predictive power than Hidden Markov Models. PCFGs are biased toward smaller trees by making them more probable than trees with many traces.

2.4.4 Decision Trees

Decision trees were first applied to language modeling by Bahl *et al.* to estimate the probability of spoken words [39]. A single node is the starting point followed by binary questions that are asked as a method to arbitrarily partition the space of histories. As the space is partitioned, "leaves" are formed and training data is used to calculate the conditional probability of $P(w|h)$ for the next element. As the traversal continues, the questioning becomes more informative by the use of information theoretic metrics. Such metrics include *Kolomogorov Complexity, entropy, relative entropy, etc* [40]. For example, if a person wants to assess how much it would cost to live in certain neighborhoods, the simplified decision tree in Figure 2.4 could be used. The root node, *Location*, is evaluated with children nodes (*Neighborhood, price.isMod*) representing values for locations. The output for these logical test are usually boolean values [41]. From Figure 2.4, we derive: "If *location=city* $\wedge$ *neighborhood=northside* $\wedge$ *condition=excellent*, then the price for homes in this area are expensive."

Decision tree algorithms are primarily composed of training data, test data, a heuristic evaluation function, and a stopping criterion function. These models use recursion from the divide and conquer data structure to induce data from the root node downward. Ultimately, decision trees represent the "gold-standard" for partition-based models. However, size, complexity and data sparseness lead to misclassification of elements when trees are large. To simplify the tree structure, Breslow *et al.* provide five top-level categories for tree simplification as seen in Table 2.2 [41].

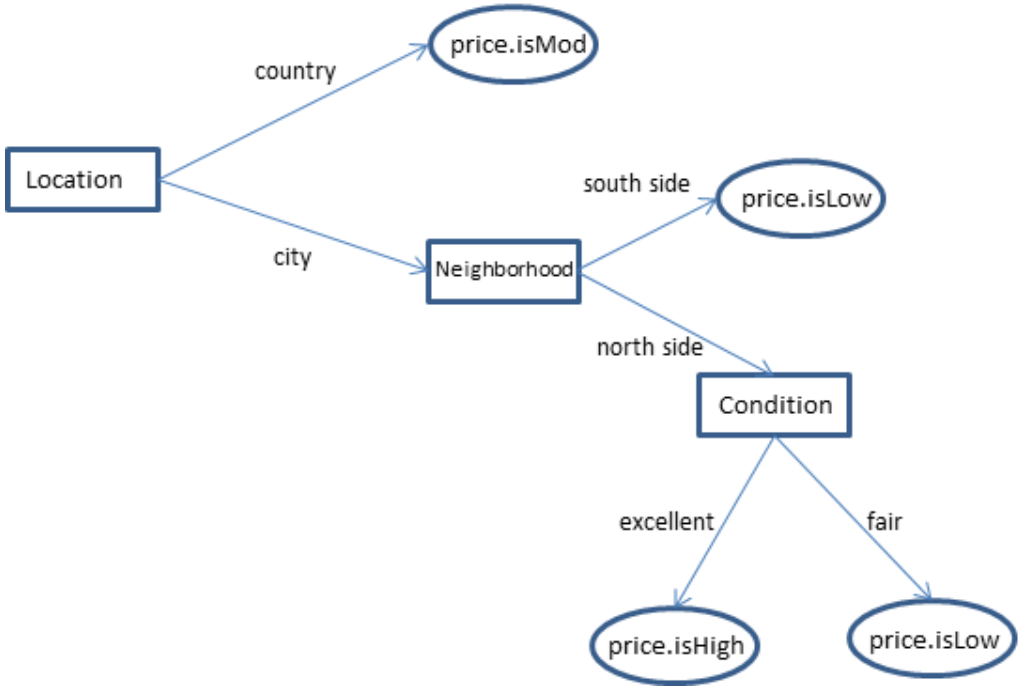


Figure 2.4: Decision Tree

Simplification Approach	**Procedures**
Tree Size Control	pre-pruning post-pruning incremental resizing
Modify Test Space	data-driven hypothesis-driven
Modify Test Search	selection measures continuous features lookahead search
Database Restrictions	case selection feature selection
Alternative Data Structures	decision graphs rules

Table 2.2: Decision Tree Techniques.

Petrovskiy uses a combination of decision tree classification and time-dependent features to learn user behavior for next-action prediction and anomaly detection from database logs generated by a banking Intranet application [42]. Results show that decision trees have fair performance for next-action prediction and very little accuracy for anomaly detection.

2.4.5 n-grams

Markov models have been heavily used for their predictive power. Markov models assume that the probability of an occurring event is dependent only on the current state of a system. As a simple example, imagine that we would like to track the probability of a Sunny (S) day or Rainy (R) day of weather. To learn the probability of a Sunny day, we could observe days for some period of time (*e.g.*, 10 days), and count the number of Sunny days observed, $nSunny$. Then, we could assign the likelihood of a Sunny day occurring to be $\frac{nSunny}{10}$. When waking up each morning, we assume that the probability of a Sunny day is given by this same fixed rate. Under this interpretation, to find $P(SSSSS)$ (i.e., five Sunny days in a row), we would simply solve $P(S)^5$. Probabilities are computed based on a set of observations. The observations can be mapped to a sequence of class labels $\{w_1, w_2, w_3, ..., w_n, ...\}$. In this sequence, w_1 represents the first observation, w_2 the second, and so on. The chain rule of probability theory computes the probability of an observation according to some prior context available at each data point. When using this method, the number of parameters grow exponentially with the number of observations in prior context. Therefore, it is not feasible to accurately estimate conditional probabilities. Instead of computing this type of probability, n-grams are used to approximate prior history by looking at the last few observations.

In most cases, it is reasonable to apply the *Markov assumption*, which assumes that the probability of observing w_i is only dependent on a very small set of preceding observations. Markov models make predictions on future elements without looking

too far in the past. Therefore, n-gram models are Markov models which use $(N-1)$ elements of context to define the current state of the model [43]. These stochastic process models are mostly stationary since we are assuming past behavior is a good prediction of what will happen in the future. However, natural language is not stationary because the probability of upcoming words can be dependent on events that are arbitrarily distant and time dependent. Therefore, the statistical models of n-grams only give an approximation of the correct distributions and entropies of natural language. Constructing or training an n-gram model requires the ability to observe example sequences occurring in the domain to be modeled. To train a model well, single events from sequences in all relevant contexts must be observed.

Studies reveal that low-order Markov models do not make accurate predictions because it does not look back far enough to past observations. Contrarily, many limitations exist for higher order models. Reduced coverage, high state-space complexity, and overall prediction accuracy are a few of the short comings of this approach. In response, Desphande *et al.* introduce a technique to combine different order Markov models to lower the state-space complexity while increasing coverage and accuracy for web pages [44]. Their selective Markov model uses frequency, confidence, and error pruning to discard certain states across different order Markov models. Manavolglu *et al.* use a combination of maximum entropy mixture models and Markov models to model behavior of web users [45]. Both models have various strengths and weaknesses but when combined they are able to accurately identify and visualize specific user behavior patterns. In this work, we seek to understand the limitations that exist for high-order and low-order Markov (i.e., n-gram) models

for user identification and the detection of outliers.

2.4.6 Hidden Markov Models

Hidden Markov Models (HMM) are Markov models whose process transitions between states in an unobservable path. However, output that is dependent on these states are visible. Basic theory behind HMM was published in a series of classical research papers [46] [47] [48]. HMMs have been conventionally applied to problems that require the recovery of data sequences that are not immediately observable. These models have been used for speech recognition, gene prediction, part-of-speech tagging, activity recognition, etc.

An HMM is characterized by the following [49]:

1. N, represents the number of states, S, in the model. Individual states are denoted as $S = \{S_1, S_2, S_3, ..., S_n\}$ and the state at time, t, is q_t.

2. M, represents the number of distinct observation symbols per state (e.g., discrete alphabet size). These symbols are denoted as $V = \{V_1, V_2, V_3, ..., V_M\}$ and correspond to the physical output of the modeled system.

3. Transition probability distribution for each state is denoted as $A = \{a_{ij}\}$.

4. B, represents the observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where $b_j(k) = P[V_k \ at \ t | q_t = S_j]$.

5. π, represents the initial state distribution where $\pi_i = P[q_1 = S_i]$.

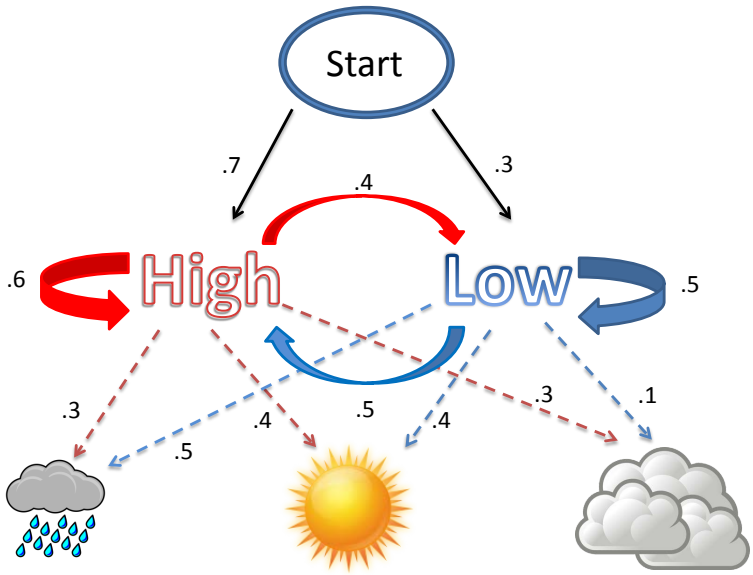


Figure 2.5: HMM for Weather Prediction

To illustrate this model, we consider the weather as a concrete example. In Figure 2.5, observable conditions are rainy, sunny, or cloudy conditions. Factors that influence theses outcomes are hidden states (e.g., high and low pressure). The *start* state illustrates the initial probability of the model. On average, the model starts with *high* pressure at 70%. Transition probabilities are represented by the change in pressure in the underlying Markov chain. In this example, there is only a 40% chance that tomorrow has *low* pressure, if today's pressure is *high*. Probabilities for observable states represent the likelihood of a particular weather condition occurring. If we observe high pressure, there is a 40% chance it is sunny. To learn tasks, HMMs must find the best set of transition states and output probabilities using a maximum likelihood estimate.

2.5 Summary

This chapter presented an overview of related work for existing continuous user authentication implementations. We explored various statistical language modeling techniques that can possibly be applied to CUA tasks. Table 2.3 presents a high-level summary of the advantages and disadvantages of each SLM explored in this research.

SLM	**Advantages**	**Disadvantages**
`Neural Networks`	Fast training process with inherent parallel structure. Training samples can be added or removed without extensive retraining.	Requires a large amount of memory resulting in slow execution of the network.
`Maximum Entropy`	Incorporates arbitrary knowledge sources while avoiding data fragmentation.	Can be infeasible due to computational challenges. Needs explicit normalization.
`PCFG`	Good for grammar induction. Can be learned from positive data and deal with grammatical errors.	Worst at modeling language for English than n-grams. Only observes structure not lexical co-occurrence.
`Decision Trees`	Ultimate partition-based model. Implicit feature selection.	Computationally expensive because the space of histories and space of possible questions can grow very large even with smaller data sets.
n-`grams`	Reduces the dimensionality of the estimation problem. Trigrams (n=3) are often used for very large training corpora and bigrams (n=2) for smaller sets.	Various smoothing techniques must be used to battle the sparse estimation problem.
`HMM`	Able to construct a model of the structure or process with observations only.	Requires annotated data for training sets.

Table 2.3: Comparison of Statistical Language Models

Chapter 3: Intruder Detector: A Tool for CUA

This chapter characterizes the application under evaluation (AUE). Specifically, we show the significance of modeling user behavior with n-grams. We also describe key contributions of this research. Each contribution provides the building blocks for the *Intruder Detector* tool.

3.1 Web-based User Behavior Analysis

Modeling the behavior of users is an ongoing challenge in various application domains [50]. In web applications, obtaining a better knowledge of the user and purpose for the application is essential to provide an increased level of security. Applications in this study are based on organizational information systems. Such systems can be used for decision support, knowledge management, and e-learning. It is important to continuously monitor users as they interact with these systems. Insider threat, especially for decision support systems, can harm an organization's security practice, data, and computer system.

The web AUE for this research is a fielded government training support website, User Productivity Enhancement, Technology Transfer and Training (PETTT) Online Knowledge Center (OKC), for the high performance computing (HPC) com-

munity. The PETTT activity is responsible for gathering the best ideas, algorithms, and software tools emerging from the national HPC infrastructure and making them available to the DOD and academic user community. This website is a repository of PETTT programmatic and technical information. It provides training course registration, course evaluation, information on several domain specific areas, and is a general source of information for its community of users. The system is a Java-based web application. Most pages are JavaServer Pages (JSP), with some servlets handling various tasks. The primary database utilizes Oracle 11g enterprise edition. Xwiki Framework and Apache Struts are leveraged to give users with elevated privileges the ability to manage their own content. This portion of the system runs over a MySQL version 5.6.11 database. The operating system environment is Microsoft Server 2008 R2.

Users have the option to authenticate via two methods: Common Access Card (CAC) and Yubikey. The CAC is used to identify DOD civilian employees, Selected Reserve, active duty uniform service personnel and government contractor personnel [51]. This card can also be used to gain physical access to buildings, controlled spaces, computer networks and systems. The Yubikey is a key-sized device that is inserted into a user's computer system USB slot to provide an added layer of authentication. If users choose to login via Yubikey, they must enter a username and password. Finally, they are directed to touch the Yubikey to generate a random passcode to enter the web application. Figure 3.1 presents a subset of pages that can be viewed by users of the AUE.

Each user account has one of the following associated roles; *User, Administra-*

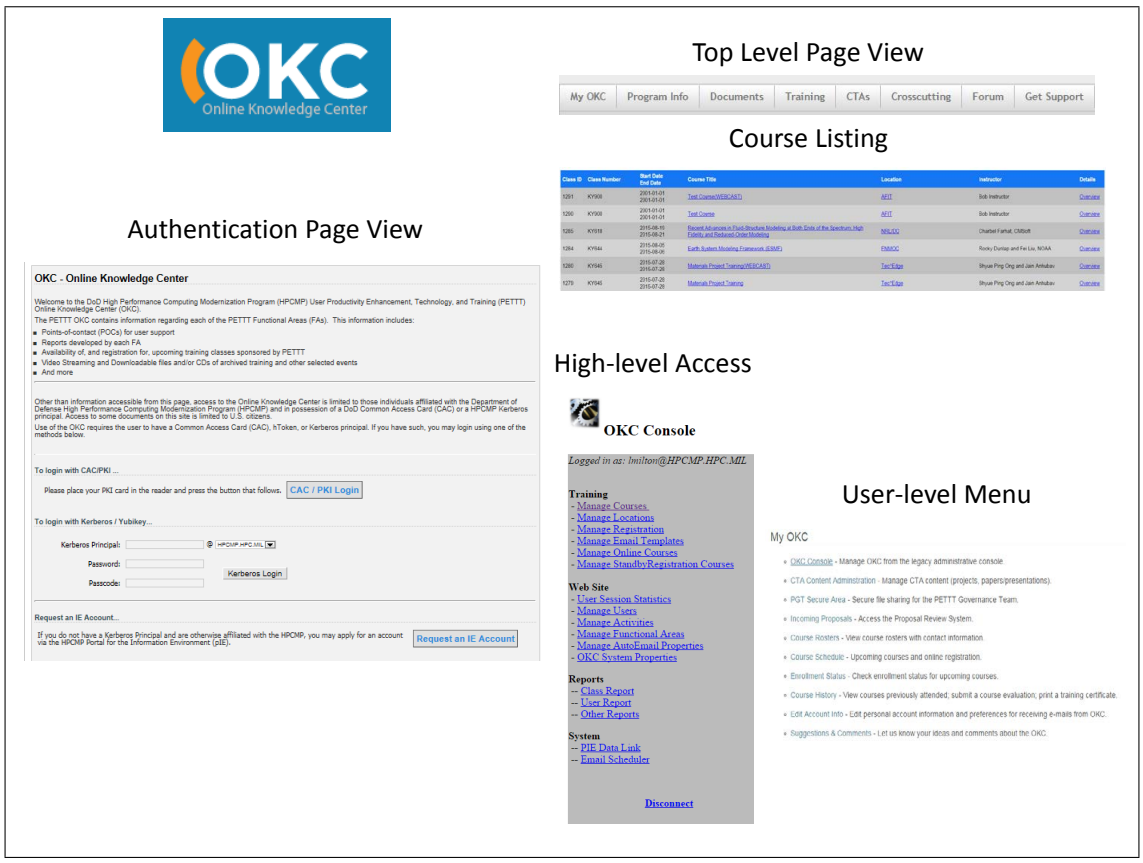


Figure 3.1: PETTT OKC Page View

tor (Admin), Management (Mgmt), Technologist (Tech). The *User* group has limited read access as well as the ability to enroll, evaluate, and register for courses. There are additional roles that provide access to areas of the system for administrative purposes (*Admin, Mgmt, Tech*). The most prominent of these is the Technologist role. These users interact with the system often as content administrators. Content administrators add updated data to the system, approve students for online training, arrange content, etc. Therefore, while we have more individual sessions available for the *User* role, the *Tech* role provides more keyword data. Users with the *Admin* role focus on technology/information integration, and data management as well as enhancements to the design and operation of PETTT OKC. The *Mgmt*

role is reserved for those who provide oversight of the application. They are not very active users of this web application. *Tech* and *Admin* roles pose the greatest risk when access is unauthorized.

3.2 Feasibility Study

To provide an assessment of our approach, we analyze four preliminary models for the classification of role-based user data by observing the accuracy of *individual actions* (IA), *frequency of individual actions* (Freq-IA), *pairs of actions* (PA), and *frequency of pairs of actions* (Freq-PA). This data is obtained from the OKC web log files. It is our intuition that accuracy from these approaches will yield less than optimal results and statistical language models, such as n-grams, must be used.

The data for each approach is partitioned into training and test set examples using the 90/10, 80/20 and 70/30 data split technique (training/test) as shown in Table 3.1. The data is sorted and placed into a training and test vector for comparison and classification. We compare the four preliminary approaches to n-gram models using the same partitioning. Section 4.1.1 provides more detail on the n-gram implementation. We run each experiment 10 times. For the evaluation of IA, we use binary numbers to identify matches in the vector. A "1" represents a keyword match and a "0" represents a mismatch. The ones are added and normalized by the size of the test set. Next, we evaluate PA. We would like to know if classification accuracy increases when pairs of keywords are used. We group training and testing sets as in the IA example using a Java HashMap data structure. If pairs are matched,

the numerical key for the HashMap is increased until the entire set is evaluated. Finally, we divide the total number of matched pairs by the total number of pairs that exist in the test keyword dataset. Next, we observe classification accuracy for Freq-IA and Freq-PA. For each individual or keyword pair, we use a HashMap to count the number of times the keyword (or keyword pair) exist in the training and test dataset. If the frequency counts of the training and testing keyword pairs are within five, we classify the frequency as a match and add it to the classification accuracy.

Data Split	**Role**	**Sessions**	**Keywords**
90/10	Management-train	247	5045
	Management-test	28	814
	Technologist-train	421	31965
	Technologist-test	47	3709
	Admin-train	521	22645
	Admin-test	58	3388
	User-train	1735	24849
	User-test	193	2808
80/20	Management-train	220	4643
	Management-test	55	1216
	Technologist-train	374	28629
	Technologist-test	94	7045
	Admin-train	463	20298
	Admin-test	116	5735
	User-train	1542	21949
	User-test	386	5708
70/30	Management-train	192	3787
	Management-test	83	2072
	Technologist-train	327	24643
	Technologist-test	141	11031
	Admin-train	405	18122
	Admin-test	174	7911
	User-train	1349	19318
	User-test	579	8339

Table 3.1: Number of Keywords and Sessions for Each Data Split

Figures 3.2, 3.3 and 3.4 show role-based accuracy results for this feasibility study. It was our intuition that the use of individual keywords would yield the worst accuracy. In each data split this accuracy is less than 10%. Classification of roles based on pairs of keywords produce the highest accuracy for each data split with less than 50% accuracy for the 90/10 data split. The Technologist role has a noticeable increase in accuracy when evaluating frequency of individual actions based on the 70/30 data split. The 90/10 data split yields the least performance for each preliminary classification technique.

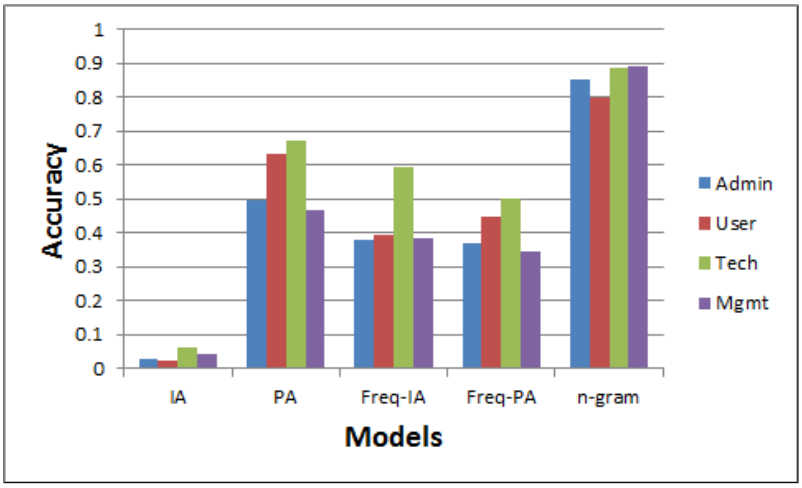


Figure 3.2: Role Classification Based on 70/30 Data Split

From this study, we show the accuracy of classifying users of organizational web-based systems based on four preliminary approaches and one SLM; *individual keywords, pairs of keywords, frequency of individual keywords, pairs of keywords, and n-grams*, using representative samples of data from pre-classified instances. Based on each data split and model, *n*-grams significantly outperform preliminary models with approximately 90% accuracy for many user roles. We hypothesize that the use of statistical language models could offer more predictive power for sequences

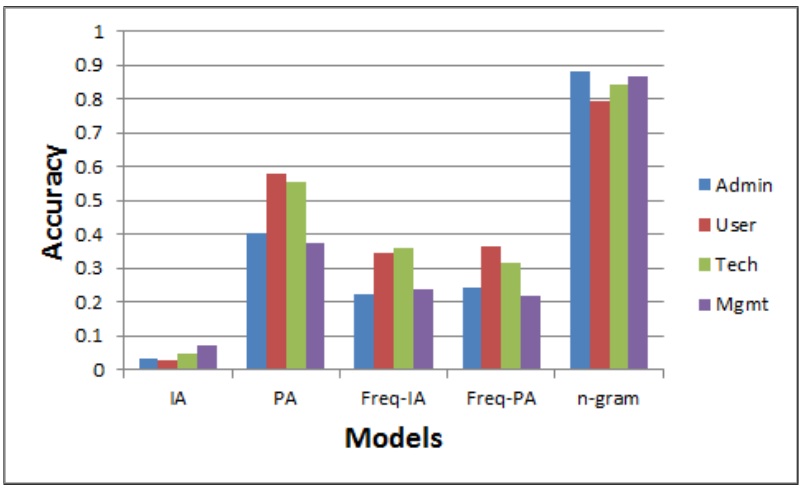


Figure 3.3: Role Classification Based on 80/20 Data Split

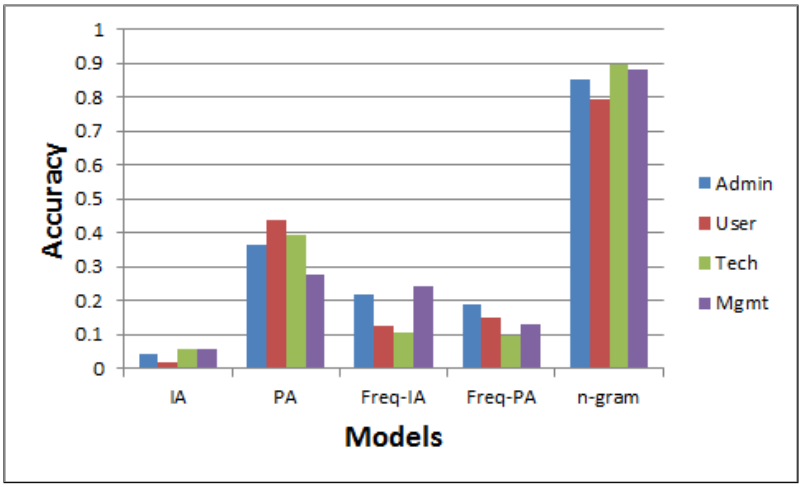


Figure 3.4: Role Classification Based on 90/10 Data Split

of web-based interactions. We address model-specific differences for n-grams in Chapter 4.

3.3 CUA Paradigm for Web-based Applications

We now discuss key contributions of this work which serve as building blocks for the *Intruder Detector* tool.

3.3.1 Contribution 1: Model Selection

Natural Language	**Web Behavior Language**	**Example**
Word	Link selection	View profile
Phrase	View	Watch online training video
Sentence	Action	Search for archived files
Paragraph	Activity	Register for course
Document	Event	Prepare course evaluation report

Table 3.2: Comparison of Natural Language and Web Behavior

It is important to understand individual and role-based behavior to detect common patterns. Specifically, we would like to use web logs to build usage profiles for each user. Grammars can be defined for human behavior and natural language [52] [53]. Table 3.2 illustrates the commonalities between natural language and web-based user behavior. Links or buttons in a web application represent a basic level of vocabulary for web behavior. A series of link clicks give the user the ability to *view* various portions of the website (e.g., course videos). Meaningful sequential link selections may lead to various *actions*, *activities*, or *events*. Since web logs represent sequential actions, they can be encoded as sequences of symbols and used with a standard NLP technique to build computational usage models [54]. In this research, we experiment with n-grams to model sequences of keywords, each of which represent a user action. We use n-grams, derived from Markov models, to understand similarity in user actions to classify the user's identity. These generative models can learn each category of users then classify the users based on the generated knowledge. This method gives us the ability to perform user authentication task

with only positive training samples.

Classification of data becomes more feasible after n-grams have been derived. Let us illustrate how n-grams can be used for classification. Suppose a bi-gram model, (i.e., $N = 2$), of user behavior that is captured after training is:

$$n = U_1U_2U_1U_3U_1U_4U_1U_5U_2U_3$$

where U_i is an observation from a user behavioral sequence. When a user's test sequence is read as $\{U_1, U_2, U_3\}$, bi-grams are generated as $\{U_1U_2, U_2U_3\}$. The probability for this bi-gram to occur will be calculated using the following equation:

$$P = \frac{\#\ of\ valid\ observations}{\#\ of\ observations} \tag{3.1}$$

The sequence is normal if P is greater than or equal to a preset threshold value, t. After applying Equation 3.1, (i.e., $P = \frac{2}{2} = 1$), the observations will be identified as normal if we select a threshold value of 0.6. We assume user activity is largely based on role (i.e., access level). In our work, probabilities of n-grams are computed based on a large web log dataset (see Table 3.1). We assume users with the same role are likely to perform similar actions. System trust increases as the user interacts with the system and no outliers are identified in the CUA user model.

Field name	Value	Description
Remote host address	134.164.78.20	IP address of client
Remote log name	-	Name is unavailable
User name	rleonard@HPCMP.HPC.MIL	User account that is accessing application
Date, time, and GMT offset	[08/April/2015:17:39:04 -0800]	Log file entry was created April 8, 2015 at 5:39 P.M. The difference between the GMT and local time is eight hours.
Request and protocol version	GET okc/admin_reg HTTP/1.0	GET command was used for the admin_reg file using HTTP version 1.0
Service status code	200	Request fulfilled successfully
Bytes sent	3401	Number of bytes sent

Table 3.3: NCSA Common Log File Entry

3.3.2 Contribution 2: Keyword Abstraction

Web server logs are obtained for system users and are grouped into sessions of activity by role. Access-level web-based systems have user roles already identified. An entry from the web server log is shown in Figure 3.5. The format for web logs was standardized by the National Center for Supercomputing Applications (NCSA), founder of Mosaic browser, Apache HyperText Transfer Protocol (HTTP), and Common Gateway Interface (CGI). Table 3.3 describes the NCSA web log entry format.

```
134.164.78.20 - rleonard@HPCMP.HPC.MIL [08/April/2015:17:39:04 -0800] "GET /okc/admin_reg.jsp HTTP/1.0" 200 3401
```

Figure 3.5: NCSA Common Log File Format

We must pre-process the logs to remove unwanted entries. This helps validate the data captured in a user's session [55]. Pre-processing web log data can be difficult

due to local caching and proxy servers. Web servers will not register repeated access to pages that are locally cached. If a proxy server is used, all web access entries will have the same identifier even though different users are accessing the website. Therefore, pre-processing is a critical component of the usage behavioral process. The following steps are used for this process:

1. *Data Cleaning*: The process of data cleaning is very important to generate an accurate picture of user activity when navigating a web application. Removing irrelevant request, data fields/columns, and system generated text are essential. For web logs, various graphics and scripts are generated which add several entries to the log file. However, in the web AUE, only JavaServer Page (.jsp) entries show user behavior and are important for logging purposes. Therefore, all entries are removed that are not related to user activity.

2. *User Identification*: We use heuristics to determine users. A user is a person that interacts with the AUE. Once the data is clean, the remaining file entries are grouped by individual user. Each interaction in the web log file has a user-id association. Once the user-ids are captured, they are classified within **ID** based on their pre-defined role in the system. We avoid identifying users by IP address because the same IP address may be used by a group of users.

3. *Session Identification*: Session identification is used to divide user accesses into individual sessions [55]. User sessions are pre-defined in the AUE web log. After checking the database for the role of each user, sessions are then grouped by role.

4. *Keyword Generation*: For each relevant web log entry, a portion of the string is captured as a keyword. We load keywords for all users/roles into a database to build usage profiles. Figure 3.6 shows an example of keywords extracted from a web log file. We do not consider parameters in this process because user entries would be too unique to categorize in a model.

Figure 3.6: Keyword Generation.

3.3.3 Contribution 3: Tool Support

ID is a tool, created during this study, to abstract keywords from log files, build n-grams, and categorize users based on observed behavior. We use customized ap-

plication programming interfaces (API), described in Section 3.4, to load input data and split the data based on pre-defined user roles or individual users. During the test phase of the experiments, a probability is assigned to a sequence of events. A probability alone does not provide continuous authentication. We use two approaches; binary and multi-class classification. Multi-class classification scores an input sequence according to one of many user models. For the mth classifier, the positive examples are all data points in class m and negative examples are all data points not in class m. Sequences of activity are categorized as belonging to the model which estimates the highest probability using the following equation:

$$u = \arg\max_{m} P(K, m) \tag{3.2}$$

A more complex scheme which can also be useful for continuous authentication is binary classification [56] [57]. For effective evaluation, this classification method requires more training and test data. This classification technique is used to judge a sequence as having likely been generated by a specific model (PASS) or not (FAIL). A probability threshold, t, is then used for this pass/fail type of judgment for a sequence. Any sequence whose probability exceeds this threshold should be considered as a PASS, $+1$, and otherwise considered FAIL, -1.

A decision rule is used to predict the class membership of a given sequence of behavioral keywords, K. When new samples are encountered, the following decision

rule is used:

$$\begin{cases} P(K,m) > t, & \text{then } y = +1 \\ P(K,m) < t, & \text{then } y = -1 \end{cases} \tag{3.3}$$

where $P(K,m)$ is the probability the behavioral keyword sequence is generated by the mth user's n-gram model. The probabilities are estimated using a training set of labeled data,

$$(m_0, y_0), (m_1, y_1), (m_2, y_2), ..., (m_n, y_n)$$

where label $y_i = \pm 1$ and depends on the class of m_i. Binary classification is used by a simple threshold and multi-class classification by comparing probabilities to translate n-gram models' estimations of sequence probability into decisions. The ability to make accurate classifications is investigated as supported by these algorithms.

3.3.4 Contribution 4: Evaluation Criteria

There is a large variation in the evaluation metrics used for classification systems. Metrics used for a research study must be appropriate for the problem domain. A confusion matrix, also known as contingency matrix, can be used to describe the performance of a classification system based on test data for which the positive (i.e., true) values are known. In Figure 3.7, a confusion matrix is used for the classification of positive and negative examples. True positives, *TP*, represent cases in which the prediction and actual value are correct (i.e., positive). True negatives, *TN*, are captured when the actual and predicted value is negative. False negatives, *FN*, represent cases where the prediction is negative and the actual category is positive.

Finally, false positives, *FP*, capture cases where the prediction is positive and the actual category is negative. False positives and false negatives are also known as *Type 1 errors* and *Type II errors*, respectively.

Optimal performance of any classification system is to reduce false positives and false negatives. Evaluation criteria, shown in Table 3.3.4, for a classifier can be

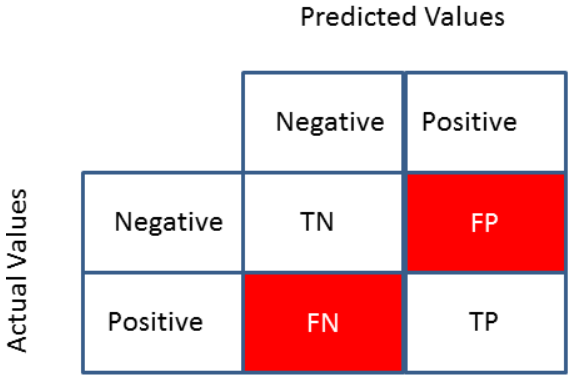


Figure 3.7: Confusion Matrix

Metric	**Formula**
Accuracy	$(TP + TN)/(TN + TP + FN + FP)$
Recall (True Positive Rate)	$(TP)/(FN + TP)$
False Positive Rate	$(FP)/(TN + FP)$
Specificity	$(TN)/(TN + FP)$ or 1 minus False Positive Rate
Precision	$(TP)/(FP + TP)$
Prevalence	$(FN + TP)/(TN + TP + FN + FP)$
F-measure	$(2\, xPrecision\, xRecall)/(Precision + Recall)$

Table 3.4: Metrics Computed from Confusion Matrix

obtained from statistical measures in Figure 3.7. To efficiently evaluate these metrics the dataset must be partitioned. We take a dataset whose class label we already know (i.e., actual data) and place them in a *training set*. To identify how well the

classifier performs with unseen data, a *testing set* is used. This will indicate how well the classifier generalizes. In this case, both sets contain previously classified data. We would also like these datasets to be large. Larger training sets provide better classifiers and larger test sets help build confidence in the evaluated metrics. We perform a repeated holdout method for the partition by randomly selecting instances for training and test sets. We use the 90/10, 80/20, and 70/30 training/test splits in this research. We repeat the hold out method 10 times and average the results to get an overall value for the metric being observed.

Accuracy is the simplest form of evaluation and is used to determine how often the classifier is correct. If the classifier correctly labels half of the dataset, then we say its accuracy is 50%. In many cases, it seems obvious that we have better predictive power as the accuracy of a model increases. Suppose we have a binary classifier to detect the presence of breast cancer in patients. From Figure 3.8,

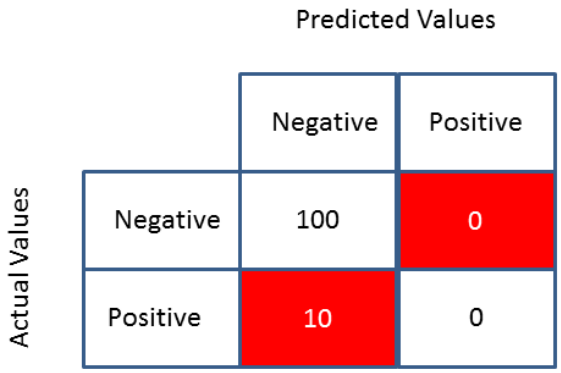


Figure 3.8: Confusion Matrix Example

$accuracy = (0+100)/(10+0+100+0) = 90.9\%$. This model is completely useless with zero predictive power because the classifier is predicting only negative examples. However, the accuracy for this classifier is very high. This scenario represents

an *Accuracy Paradox* which states that *models with a given level of accuracy may have greater predictive power than models with very high accuracy.* In some cases, accuracy alone can be misleading. This metric is better used together with other measures. Precision and recall address the imbalance that can occur in a dataset. Precision answers the following question: When the classifier predicts a positive example, how often is it correct? On the other hand, recall answers: When evaluating all the positive examples in a dataset, what fraction of the dataset did the classifier identify? A perfect classifier will have 100% precision and recall. However, in real world classification tasks, reaching this optimal performance is difficult to achieve. If the classifier is tweaked to offer very high recall, then precision rates suffer. Alternatively, classifiers that are tweaked for high precision rates, suffer from poor recall rates. To provide balance, the f-measure is used as the weighted harmonic mean of precision and recall. There are many metrics for evaluating classification systems. In this research, we consider a specific set of metrics for evaluation since various metrics provide different and valuable insight into how each model performs. False positive rates and prevalence are measured, in addition to, accuracy, recall, precision and f-measure to provide additional analysis for each model.

3.4 Continuous User Authentication Infrastructure

We use several custom APIs for the continuous user authentication infrastructure. The WeblogDataHandler is a front-end API that is used to load keyword data and split it into training and test sets. Table 3.5 provides details for each method call.

TestDataManager is a Java tool used for storing and retrieving keyword data (see Table 3.6). We use a non-relational document database, MongoDB[1], as a back-end data store. MongoDB is a NoSql (i.e., non-relational) database structured in favor of JavaScript Object Notation (JSON) documents. We serialize each keyword document into JSON before adding them to the MongoDB for storage.

We use the `BerkeleyLM` N-gram Language Model Library to construct n-gram models from keyword data. `BerkeleyLM` is a library for estimating and storing large n-gram language models in memory [58]. This library also provides efficient access to data. We extend this open source Java library with the TestDataManager and Analyzer APIs. Tables 3.7 and 3.8 show all methods used for these extensions. Specifically, we use the `categorizeStoredSequence` method for multi-class classification and the `acceptRawSequenceGivenModel` method for binary classification. Each API is designed to handle large-scale experiments with keyword data. Figure 3.9 provides a detailed high-level view of the CUA framework. This figure outlines the steps we follow to complete this task.

3.5 Intruder Detector in Action

We will now show, via an example, how **ID** works when web log data is present. Assume that, over time, web logs have been collected for users of a web-based system. **ID** has developed, for each user, an n-gram model, essentially representing the user's typical behavior. Alice, an administrator, typically starts from her user profile page to connect to the admin page. From the admin page, she can perform a

[1]http://www.mongodo.org.

Method	Description
`void splitLargeDataForScheme( String scheme)`	Split abstracted keyword data
`boolean categorizeSessions( String scheme, int modelOrder)`	Categorize n-grams by role on the same data that trained models
`boolean ngramsAgainstOthers( String scheme, String myGroup, int modelOrder, int ngramOrder, float threshold, boolean mine)`	Returns n-gram categories when using training and test data from various groups
`float categorizeStrings( (String scheme, int modelOrder, List<String> stringstoCategorize, List<String> actualGroups, String method, float threshold, String modelChoice)`	Returns the n-gram probability based on the binary or multi-class categorization method

Table 3.5: WeblogDataHandler: Front-end API for Training and Test Data Categorization

Method	Description
`boolean clearTestSuite(String dbId, String suiteId)`	Clear the suite before saving new keywords
`boolean addArtifactToTest( String dbId, String testId, String artifactPath, ArtifactProcessor`	Save the keyword file for each user/role to the database
`boolean addTestToSuite(String dbId,String testId,String suiteId)`	Add keywords with a given ID to a suite with given ID

Table 3.6: TestDataManager API: Storing and Retrieving Keyword Data

Method	Description
`String categorizeStoredSequence( String test, List<String> suiteId, int order ArtifactProcessor processorClass, boolean useWrapperSymbols)`	Categorizes stored sequence for multi-classification approach
`String acceptRawSequenceGivenModel( String test, List<String> suiteId, int order ArtifactProcessor processorClass, boolean useWrapperSymbols, float threshold)`	Categorizes stored sequence for binary-classification approach using threshold value

Table 3.7: Analyzer API: Intermediate n-gram Model Computation

Method	Description
`ArrayEncodedProbBackoffLm<String> computeModel( String dbId, String suiteId, int maxOrder)`	Compute an n-gram model from a given test suite with a given order, returning the result as a BerkeleyLM object
`ArrayEncodedProbBackoffLm<String> getNgramModel(String dbId, String suiteId)`	Return the n-gram model currently associated with a test suite

Table 3.8: TestDataManager API: n-gram Models

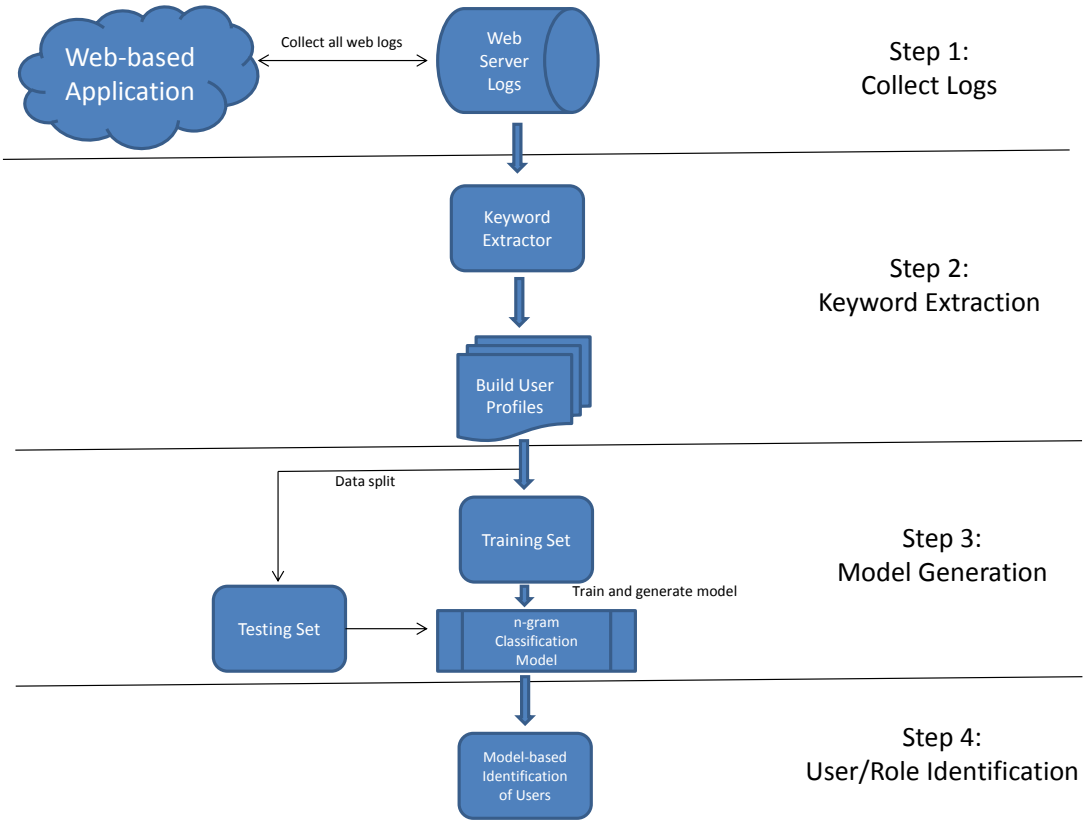


Figure 3.9: CUA Framework Description

set of tasks associated with this role, namely submit new bugs, view existing bugs, submit new change requests, view current change requests, etc.

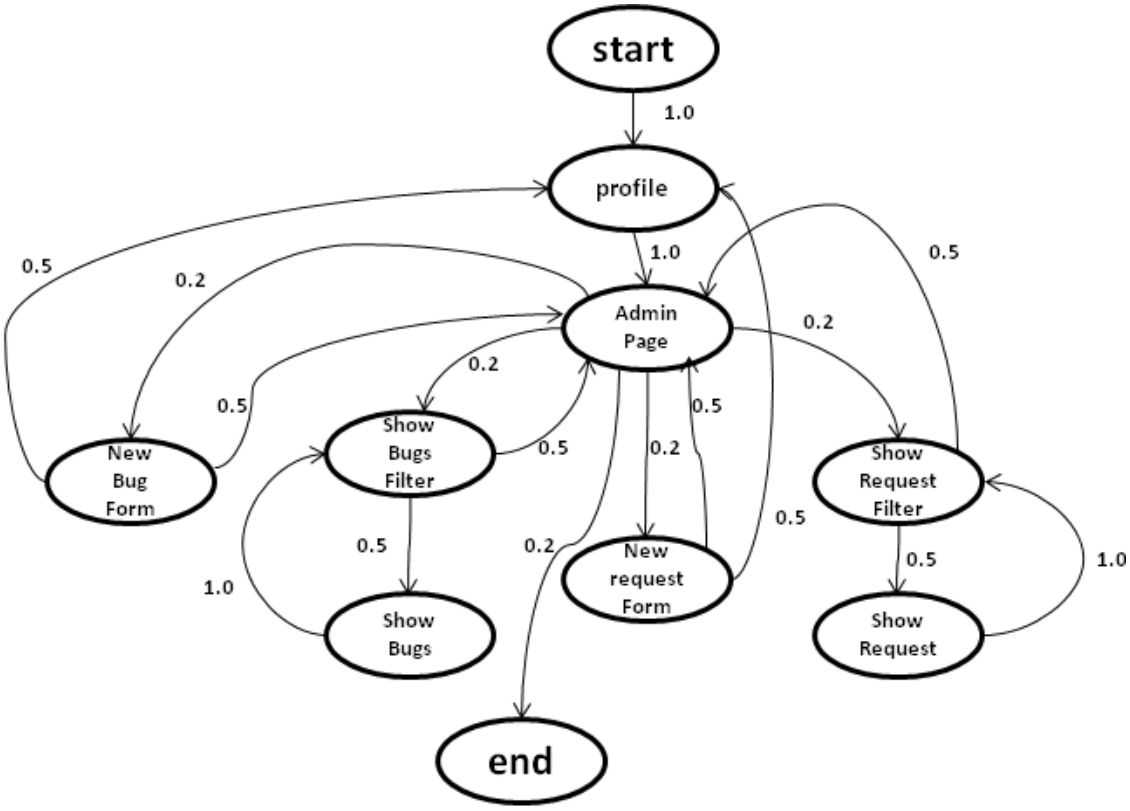


Figure 3.10: n-gram Model of Alice.

Figure 3.10 represents an n-gram behavioral user-model for Alice. The nodes in the model represent actions that Alice performs on the web pages. Edges represent a workflow relationship between these actions; the numeric values represent the probability that these actions will occur. For example, a probability of 1.0 from "start" to "profile" implies that Alice always starts at the "profile" page; once in the "admin" page, the user is likely to start a "new bug form" with the probability of 0.2. Once in the "new bug form," Alice has the option to go back to the "profile page" or "admin page" (with equal probability); the user may then return to the

"new bug form." However, there is no option to go directly to the "show bugs filter" page from a "new bug form."

ID has a stored model for Alice since she has used the web application in the past. However, **ID** is continuously monitoring her. As she interacts with the software, a series of web requests, in the form of web log entries, are recognized by the tool. These web log entries are translated into a sequence of keywords: ⟨*start, profile, admin page, show bugs filter, show bugs, show bugs filter, admin page, end*⟩. **ID** continuously checks all the stored models and matches the sequence to Alice's model. Alice is authenticated and allowed to continue using the application.

After some time, Alice decides to take a break and walks away from her computer before logging out of the system. Bob, without authorization, takes control of her session. Bob is a user with lower system privileges and has no idea how to work the system as an Admin. The keywords generated from Bob's interactions are ⟨*admin page, show request filter, admin page, new bug form, admin page, end*⟩. When comparing this sequence to the usage profile for Alice, **ID** identifies Bob as a potential intruder and the session is ended (or Bob is asked to re-authenticate, depending on how **ID** is configured).

There are several points to note regarding this overview scenario. First, we deliberately keep the n-gram model very simple so as to be easily understandable. This simple model represents N=2 with a history of zero. At this step, it is important to understand that the n-gram model captures much more information than is apparent with this simple example. Second, we show only one model (e.g., Alice's model). In general, **ID** will maintain multiple models, one for each user. Having a

model for Bob may give **ID** the ability to capture his identity. However there are some limitations to pinpointing Bob's identity which will be discussed in this work. Third, such models may be maintained for classes of users, not just individual users. It is important to distinguish between multiple user types and their different behaviors (e.g., admin users typically access the "take tape backups" page; conventional users do not).

Finally, **ID** can be configured to incrementally improve its models during execution. Consider that Alice executes a new sequence of actions not contained in her n-gram model. **ID** will recognize the mismatch and ask for re-authentication (e.g., via a password or supervisor approval). If this transaction is successful, then the sequence of actions that was not recognized is added to the n-gram model, thereby improving its accuracy. This will also be the case if the user interface is updated. Subsequently, if Alice performs the same sequence of actions, **ID** will recognize it as legitimately belonging to Alice. When fielded, we envision **ID** to be used in a training mode to build baseline models of all users; and then used in a deployment mode.

3.6 Summary

In this chapter, we described the main contributions of this work. We used a practical implementation of the n-gram model to estimate the relative likelihood of a sequence of web log keywords. We provided a typical use-case for the *Intruder Detector* tool. Novel techniques for keyword abstraction, statistical language modeling,

and optimal classifier evaluation are the key building blocks of this web-based CUA approach. In the next chapter, we provide details of an empirical evaluation of our implementation.

Chapter 4: Empirical Evaluation of Continuous User Authentication

This chapter provides a detailed description of a set of experiments using real-world user data designed to investigate the utility of n-gram models for continuous user authentication. Specifically, we answer three research questions using a set of classification-based performance metrics.

4.1 Performance Assessment

We conduct performance assessments to evaluate the feasibility of our approach. A variety of performance metrics are available for classification systems. We start our analysis with the following metrics:

- **Accuracy:** *How often is the classifier correct?*
- **Recall (i.e., True Positive Rate or Sensitivity):** *When it's actually X, how often does the classifier predict X?*
- **Precision:** *When the classifier predicts X, how often is the classifier correct?*
- **F-measure** *Harmonic mean of precision and recall.*
- **False Positive Rate:** *When its actually not X, how often does classifier predict X?*

- **Specificity:** *When it's actually not X, how often does the classifier not predict X?*

- **Prevalence:** *How often does each role/user occur in the data set?*

Based on these metrics, we will answer the following research questions:

- RQ1: *Can discriminating user models be constructed to determine user types?*

- RQ2: *Can the model recognize various legitimate users who are operating in the same user session?*

- RQ3: *Can usage profiles be used to identify outliers in the user's behavior?*

4.1.1 Experimental Setup

Once the keyword abstraction technique outlined in Section 3.3.1 is complete, we begin predicting user behavior. When predicting individual and role-based user behavior, web logs are filtered to abstract only those users who have at least two sessions of activity. To ensure we have enough keyword data for individual user behavior, we abstract only those users who have at least 800 keywords. Tables 4.1 and 4.2 provide the number of users and roles that were captured after the keyword abstraction process. We also identify how often labeled data occurs in our dataset using the prevalence metric. Experiments were conducted using various configurations to better understand the strengths and weaknesses of our approach. We split the data, based on sessions of activity, using the 90/10, 80/20, and 70/30 training/test data split for individual users and roles using the same number of

keywords and sessions described in Section 3.2, Table 3.1. Data splits help maintain purity of the evaluation. Therefore, no model should be evaluated on its ability to classify data which was used during its own training phase. The models for each data split are based on $N = 2$ (bi-grams), $N = 3$ (tri-grams), $N = 4$ and $N = 5$. The use of data splits with various n-gram models will identify which model performs better for the continuous user authentication task. We use the repeated holdout method to randomly select training and test sets. We run 10 iterations for each model and provide an average value for each metric.

Role	**No. of Users**	**Prevalence**
User	3893	14%
Admin	9	18%
Management	41	9%
Technologist	2	59%

Table 4.1: Role-based Dataset

Role	**User ID**	**Prevalence**
User	User3	11%
Admin	User9	6%
	User6	13%
	User10	7%
	User8	8%
	User4	9%
Management	User1	5%
	User7	2%
Technologist	User5	22%
	User2	17%

Table 4.2: User-based Dataset

Constructing, or training, an n-gram model requires the ability to observe example sequences occurring in the domain to be modeled. To train a model well, single events must be observed from sequences in all relevant contexts. This is the

challenge that occurs when evaluating classifiers in the NLP domain. Conditional probabilities are captured when evaluating n-gram models. The number of possible parameters needed by the model grows not only according to events observed, but also exponentially by the history being considered. For the best possible model, our training phase requires observations of every possible event in every possible context. Additionally, there is a need to observe each context multiple times to have confidence that parameters are accurate. Without sufficient training, event sequences may be encountered during the test phase for which there is no knowledge to provide a probability estimate.

Researchers from the NLP domain have addressed this challenge of insufficient or sparse data for n-gram models. A concept known as *smoothing* involves saving some probability mass in an n-gram model's probability distribution for unseen events [59]. Several smoothing techniques exist for language models such as Additive, Good-Turing Estimate, Katz, Witten-Bell, and Kneser-Ney [59]. The models in this experiment use the Kneser-Ney smoothing technique [60]. Smoothing has significant effects on the probability of n-gram models. At a high level, Kneser-Ney smoothing involves:

- Reserving probability mass for unseen events by reducing all probabilities by a constant discounting percentage.
- Estimate missing higher-order conditional probabilities by incorporating the observed frequencies of lower-order prefixes (a concept known as *backoff*).
- More precisely, combining the frequency of lower-order n-grams with a concept

called *continuation probability*, which estimates the probability that an event completes an n-gram, to estimate the probability of event sequences.

The original reference should be reviewed for a complete explanation of Kneser-Ney smoothing [60]. Potential risks of applying Kneser-Ney to the domain of events carried out on software are violations of key features above. For example, if very few events are being performed by users, applying a constant discounting to every probability may save too much probability mass for unseen events.

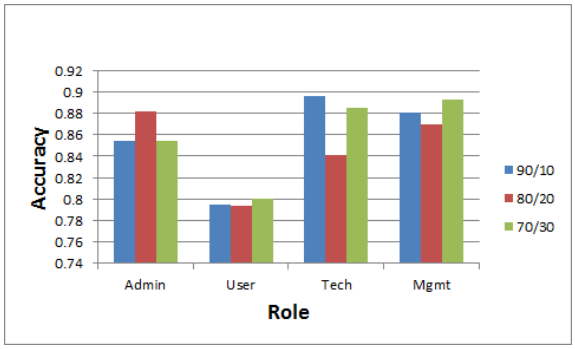
4.1.2 Experimental Results and Analysis

4.1.2.1 Multi-class Classification

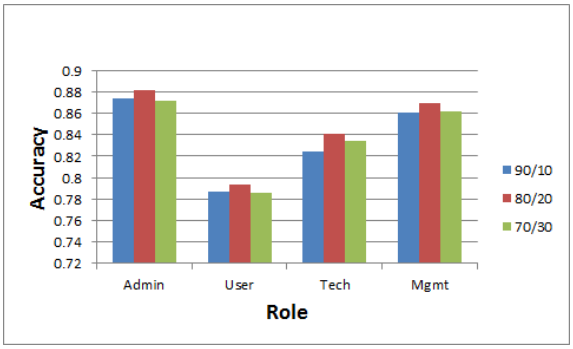
For each metric, we compare **ID**'s recommended category to the actual category of each test session. A percentage of keywords are reserved based on a data split to represent the testing set, E, and use the remaining data as the training set, R, to train an N order n-gram model for the specified class. $P(K, m)$ is calculated for each sample of keywords, K, from E. This represents the probability that the keywords are generated by the mth users n-gram model. Finally, m is selected with the highest probability for the behavioral keyword sequence, $P(K, m)$, and used as the prediction value.

We compute each metric for each class label and analyze the performance. We also average these values to get an overall metric value. We first consider whether unique profiles can indeed be constructed. Models are developed for each user and role as described in Section 3.3.3. To evaluate role-based behavior for RQ1, we

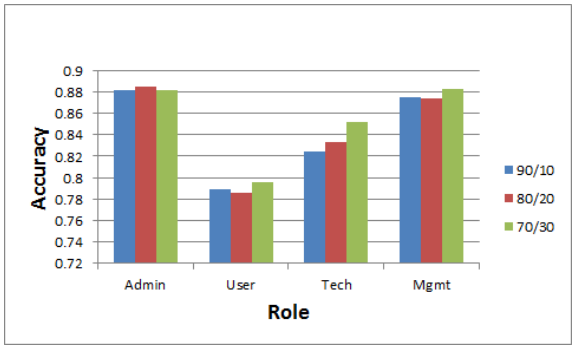
observe that a random model with no observations would achieve 25% accuracy when categorizing users. If the models generated are effectively capturing unique details about the behavior of users within roles, we would expect to see much greater overall accuracy in multi-class classification.



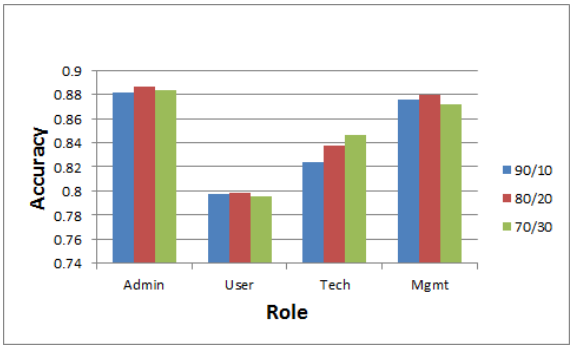
(a) N=2



(b) N=3



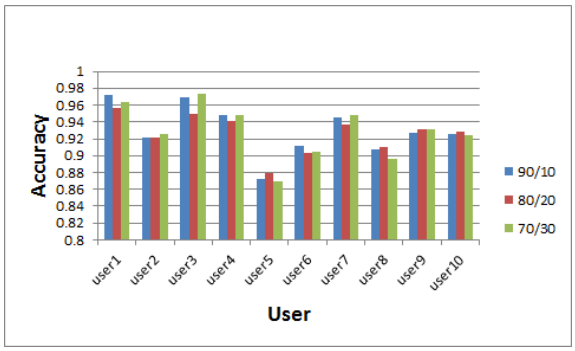
(c) N=4



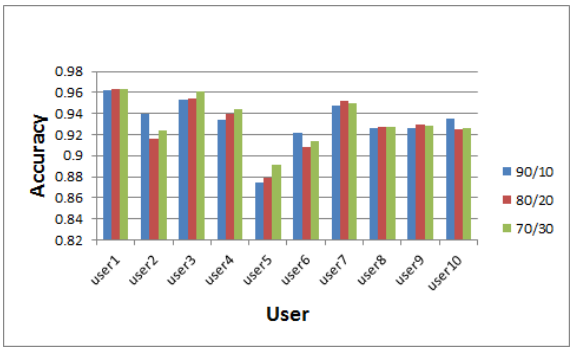
(d) N=5

Figure 4.1: Role-based Accuracy

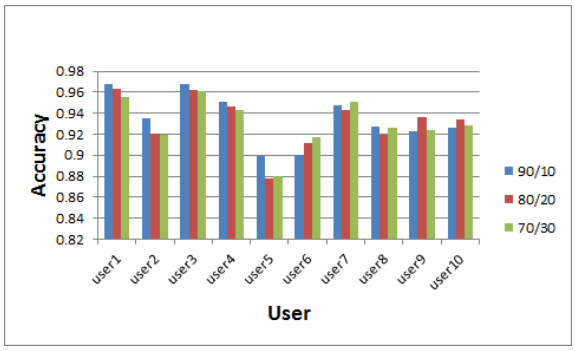
Instead of focusing on the four pre-defined user roles, RQ2 focuses on the ability of n-gram models to capture the behavior of specific users regardless of role. To evaluate this research question, the data is filtered to include only those users which have at least two sessions of activity and at least 800 total keywords. For the users meeting this criteria, we train models according to the n-gram approach. After



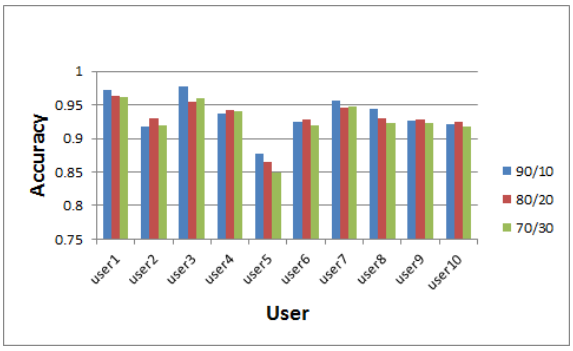
(a) N=2



(b) N=3

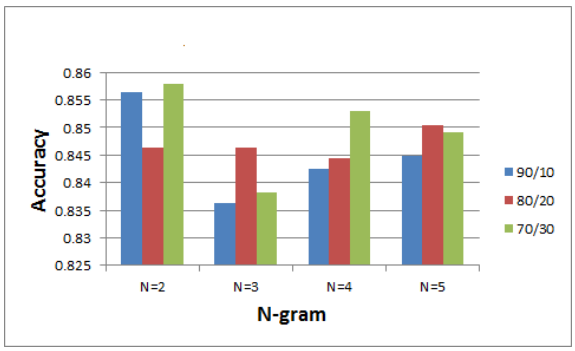


(c) N=4

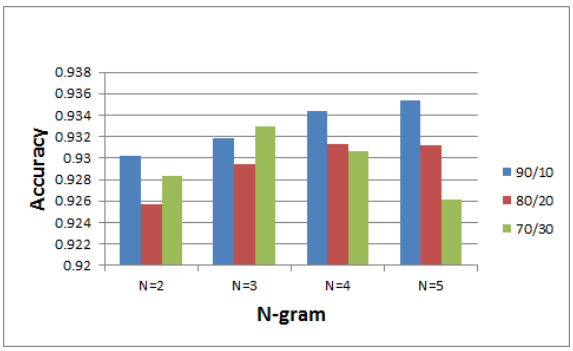


(d) N=5

Figure 4.2: User-based Accuracy



(a) Overall Role-based Accuracy



(b) Overall User-based Accuracy

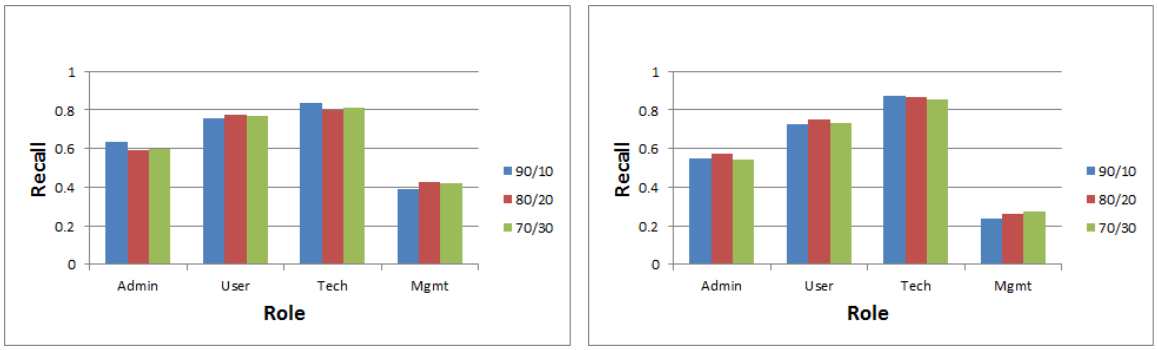
Figure 4.3: Overall Accuracy

filtering the data, 3945 users are captured for role-based classification and ten users are captured for user-based classification. Accuracy shows how often the classifier

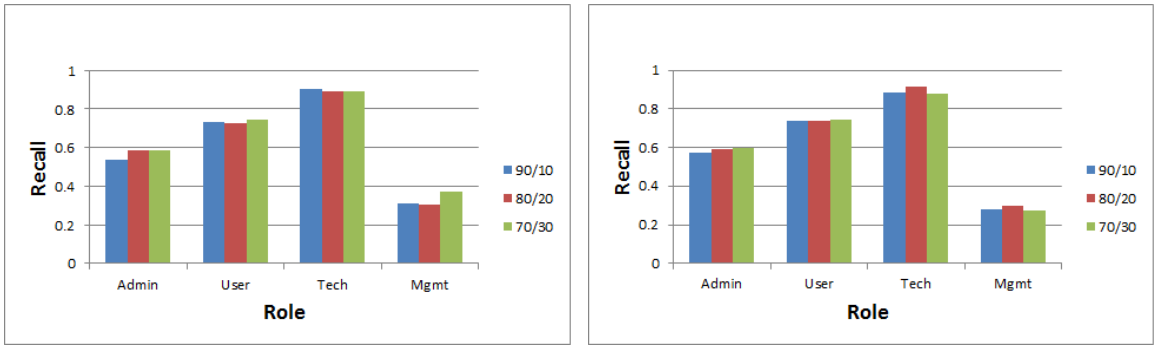
is correct when classifying roles or users.

$$accuracy = \frac{(TP + TN)}{total} \tag{4.1}$$

where *total* is the total number of predictions made for a particular user/role. Accuracy results are shown for roles and users in Figure 4.1 and Figure 4.2. We average the role and user based accuracies and present them in Figure 4.3.



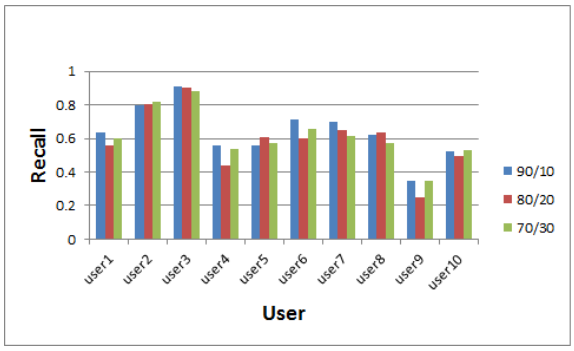
(a) N=2 (b) N=3



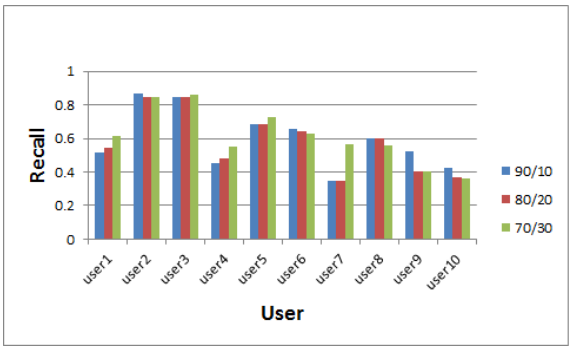
(c) N=4 (d) N=5

Figure 4.4: Role-based Recall Rate

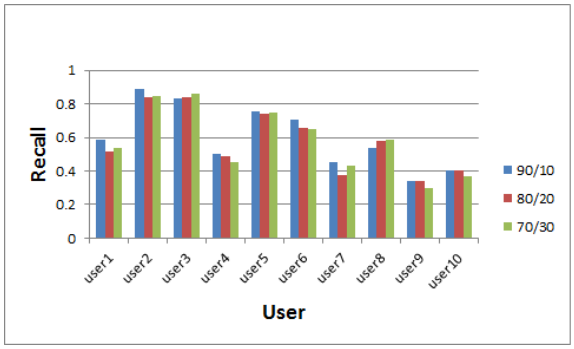
There are 63,467 user-based and 95,223 role-based keywords generated for this study. Based on our results, the 90/10 data split shows linear increase of accuracy for



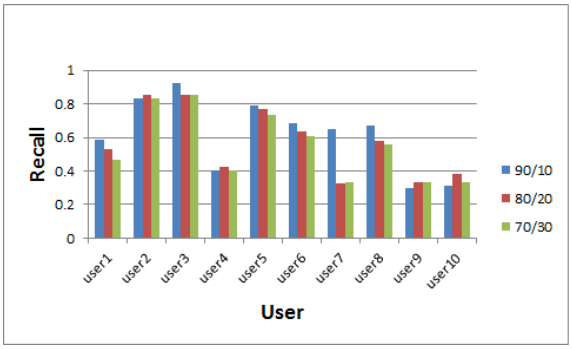
(a) N=2



(b) N=3

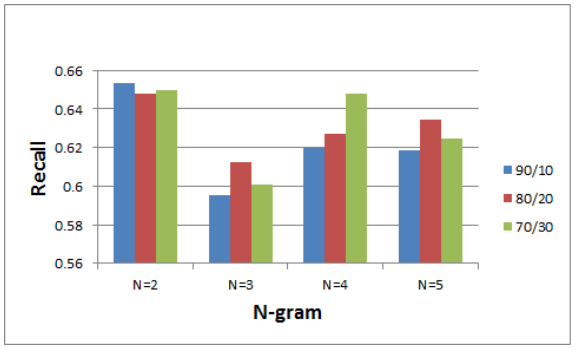


(c) N=4

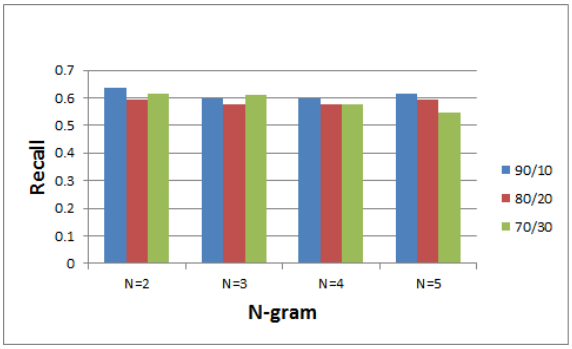


(d) N=5

Figure 4.5: User-based Recall Rate



(a) Overall Role-based Recall



(b) Overall User-based Recall

Figure 4.6: Overall Recall

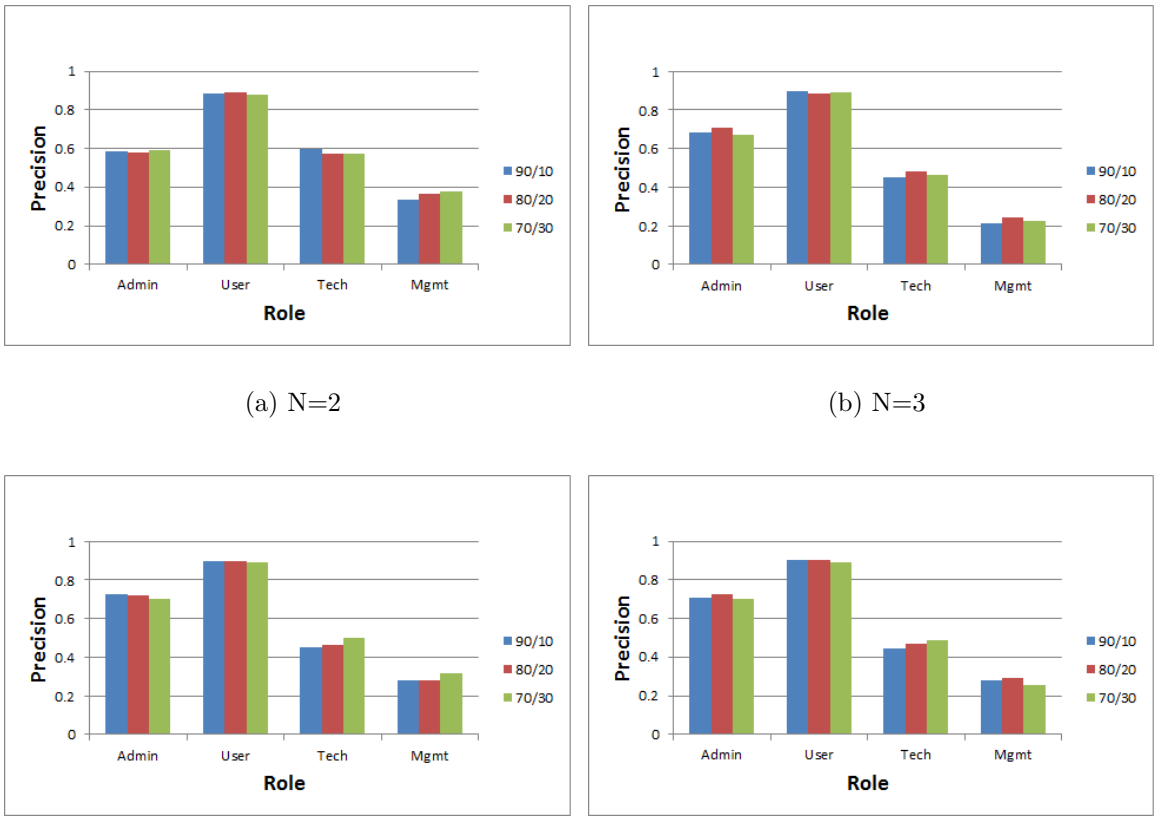
smaller training sets (individual users) with approximately 93-94% accuracy (Figure 4.3b). For larger datasets (i.e., role-based), accuracy is less than 90%. On average,

the best role-based model of accuracy is N=2 at approximately 86% (Figure 4.3a). Accuracy of the system generally decreases when classifying a large number of users in the role-based experiment. Overall, each role-based model achieved above 78% accuracy. The Tech role has the largest number of keywords used for training each data split. Initially, we expected this role to have the highest accuracy for each model and data split. However, the Tech role performed best (i.e., 90%) for bi-grams with the 90/10 data split. Conversely, the User role has the largest number of training sessions and lowest accuracy (less than 80%) for each model. The User role has 3,893 users in the dataset with short sequences of interaction. Because these sequences exist, the n-gram model may have difficulty learning behavior from this category of users. From equation 4.1, we see that accuracy focuses more on the total number of *True Positives* and *True Negatives*. From this metric, we are unable to evaluate false classifications. We also observe a large class imbalance based on the labeled dataset that can possibly lead to an *Accuracy Paradox*, described in Section 3.3.4. Therefore, we have minimum confidence in the accuracy metrics presented. This leads to the evaluation of recall, precision, f-measure, and false positive rate.

Next, we evaluate each model's *recall*. Recall measures the proportion of positive examples that are correctly identified as positive. This measure of relevance uses the *True Positive* and *False Negative* values for a class of users.

$$recall = \frac{TP}{TP + FN} \tag{4.2}$$

Figures 4.4 and 4.5 show recall rates for roles and individual users, respectively. The



(a) N=2

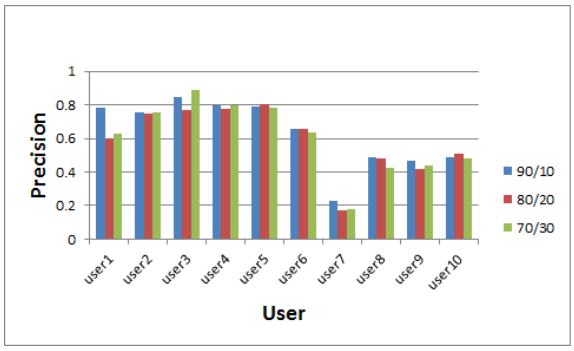
(b) N=3

(c) N=4

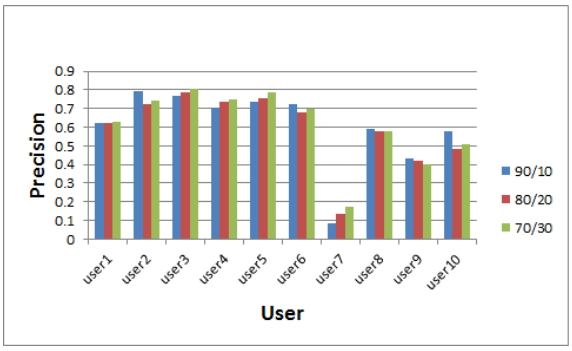
(d) N=5

Figure 4.7: Role-based Precision Rate

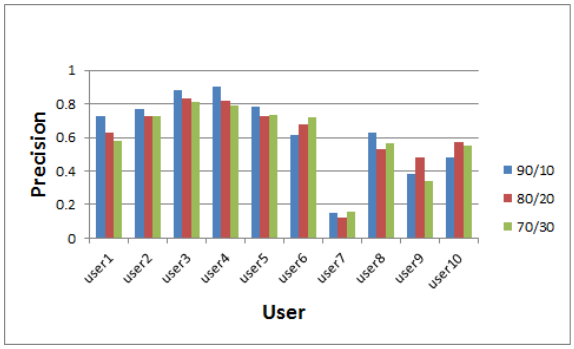
Tech role's recall increases approximately 10% as N increase from 2 to 3 through 5. The User model, with more users and less keywords, performs best for bi-grams at approximately 75% (Figure 4.4). The Mgmt role has the least recall due to the small number of sessions and keywords used for training and testing. However, we observe higher recall for individual users within the Mgmt role (e.g., User1, User7) using the bi-gram 90/10 data split (Figure 4.5-a). We assumed User5, from the Tech role, would have the highest recall since this user has the largest number of training keywords for each data split. Yet, we observed approximately 60% recall for 90/10 bi-grams and an increase to 80% recall as N increased to 5. This user is one of the



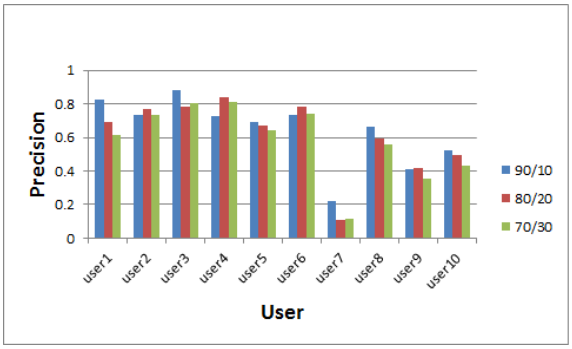
(a) N=2



(b) N=3

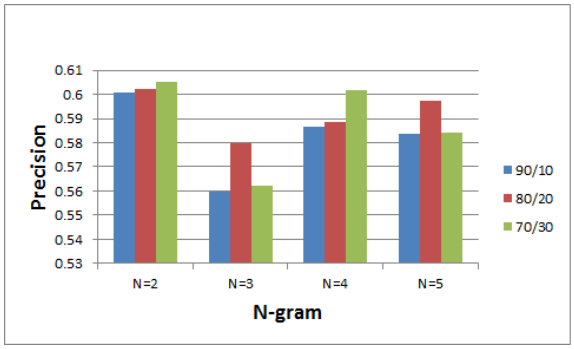


(c) N=4

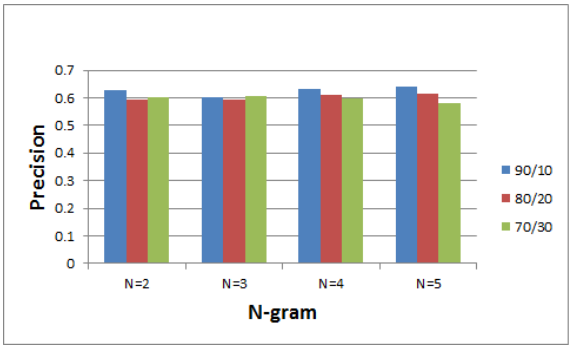


(d) N=5

Figure 4.8: User-based Precision Rate



(a) Overall Role-based Precision



(b) Overall User-based Precision

Figure 4.9: Overall Precision

most active web users. We consider the classification task performance struggles due to a high variability in the daily task of this particular user. User-based models

show a steady overall recall rate for each data split and model at approximately 60% (Figure 4.6-b). However, this steady rate is not observed for larger session/keyword data sets (i.e., role-based). In this case, the bi-gram 90/10 model has the highest overall recall rate at approximately 65% (Figure 4.6-a).

Precision measures the fraction of predicted examples that are relevant. This metric considers *True Positive* and *False Positive* values.

$$precision = \frac{TP}{TP + FP} \tag{4.3}$$

We notice a shift in performance when evaluating precision. The User role consistently has the highest precision rate for each model at approximately 90% with less than 60% precision for the Tech role. This observation holds for individual users within this role (e.g., User3) with approximately 90% precision for the 70/30 bi-gram model. Therefore, the classifier is more precise when predicting roles that have more per-session data available. The lack of precision for User7, from the Mgmt role, is due to the small number of sessions and keywords captured for this user. Figures 4.7 and 4.8 show precision rates for roles and individual users, respectively.

Historically, precision and recall have been used as classification performance metrics. In many machine learning research papers, precision is more informative for non-binary classifiers. During our analysis, we determined which metric or combination of metrics is most informative for the task of identifying users and roles. In some cases, it is very difficult to assess the performance of precision and recall separately. The f-measure, as described in Section 3.3.4, combine these into a single

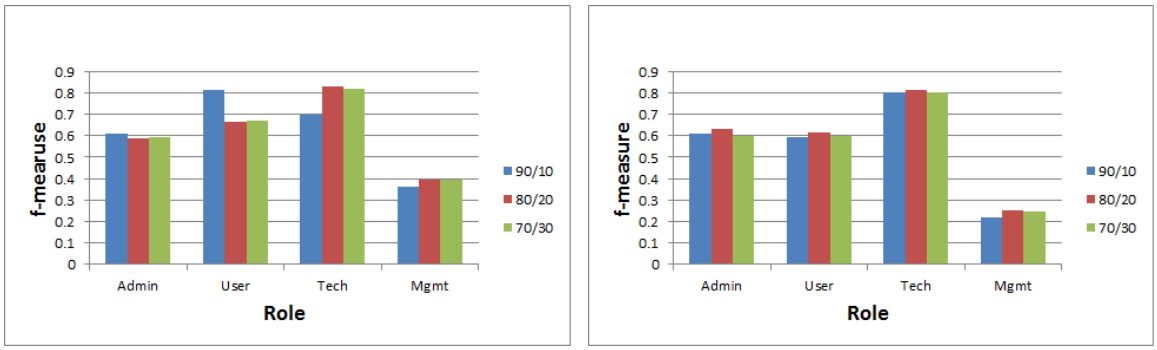
measure of classification effectiveness. Figures 4.10 and 4.11 show f-measure results for roles and individual users, respectively.

$$f\text{-}measure = 2 \times \frac{precision \times recall}{precision + recall} \tag{4.4}$$

Next, we measure the false positive rate (FPR). This metric is used to determine the rate at which false alarms are generated. Optimally, we would like this ratio to be very low. FPR becomes critical when users with lower privileges are predicted to be users with higher privileges. This action can cause severe risk to a system. Results for the FPR are shown in Figure 4.13 and Figure 4.14 for roles and individual users.

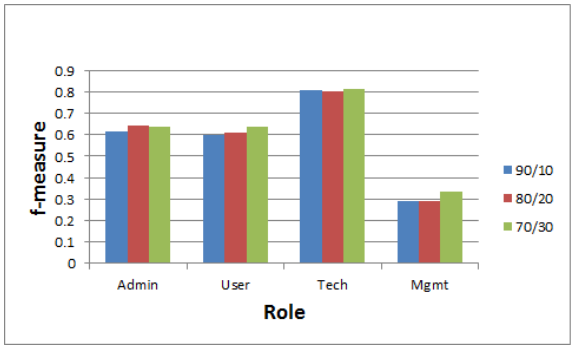
$$FPR = \frac{FP}{TN + FP} \tag{4.5}$$

The f-measure for the User role is 81% with 14% false positive rate (90/10 data split only) for bi-grams and decreases to 60% for the rest of the models (N=3 through N=5). Since this category of users has the largest number of training sessions, bi-grams may have better performance when more per-session behavior from each user is present in the dataset. Individual users with the User role have a high f-measure (87%) with the same model (bi-gram 90/10 data split). In this case, recall and precision are above 70% (precision=88%, recall=75%). User3, from the User role category, has the highest f-measure among all users individually evaluated. If we only rely on accuracy, our results would tell a different story. From the accuracy perspective, the User role has the lowest accuracy for each model.

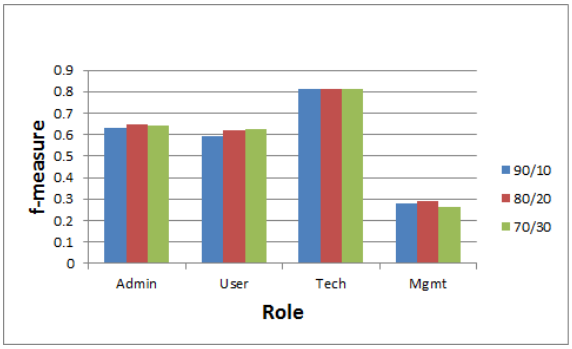


(a) N=2

(b) N=3



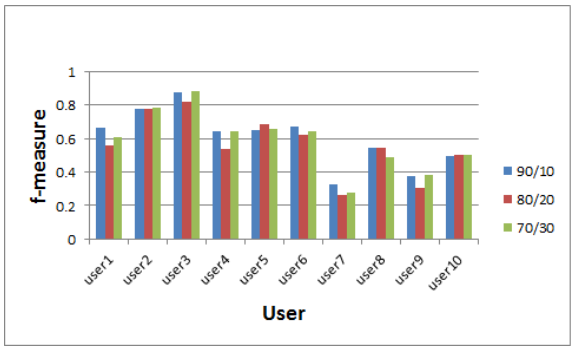
(c) N=4



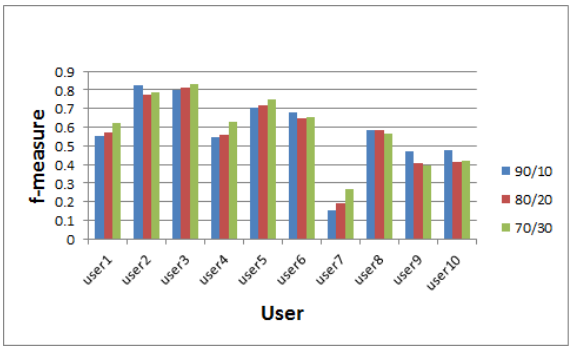
(d) N=5

Figure 4.10: Role-based F-measure

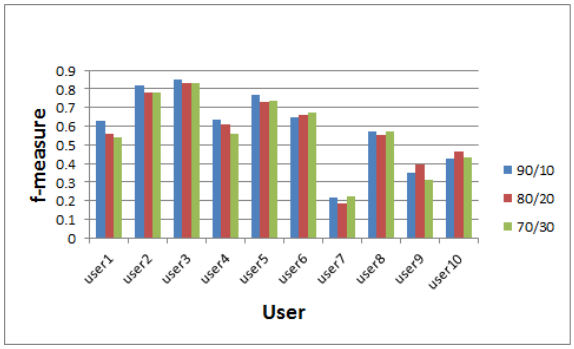
The f-measure for the Tech role is approximately 83% for bi-grams with the 80/20 and 70/30 data splits (with 10% FPR) and holds at a steady rate of approximately 80% for other models and data splits. The Tech role has the largest number of keywords used for training with 59% prevalence. Generally, the larger the training set, the better the classifier for this category of users. The Mgmt role has the lowest f-measure for each model. All data splits for this category perform below 40%. This role has the smallest number of training sessions and keywords in our dataset. The Mgmt user, User1, has a bi-gram, 90/10 data split, f-measure of 67% and Mgmt user, User7, has a bi-gram, 90/10 data split, f-measure of 30%. After



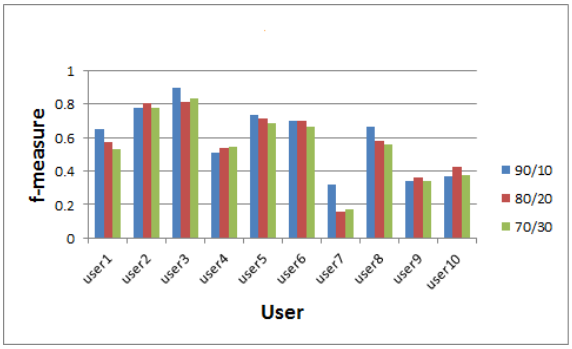
(a) N=2



(b) N=3

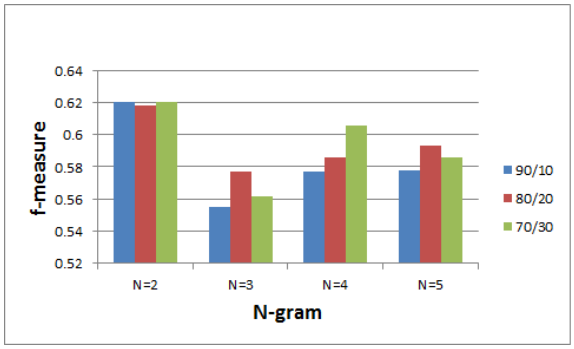


(c) N=4

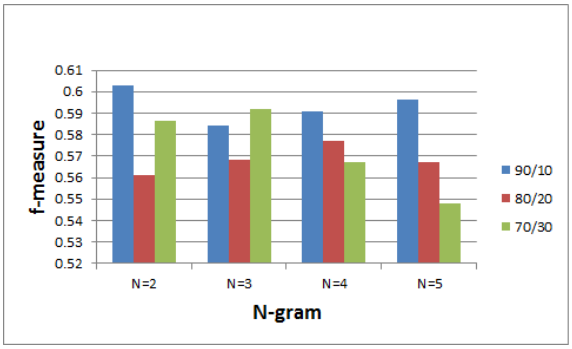


(d) N=5

Figure 4.11: User-based F-measure



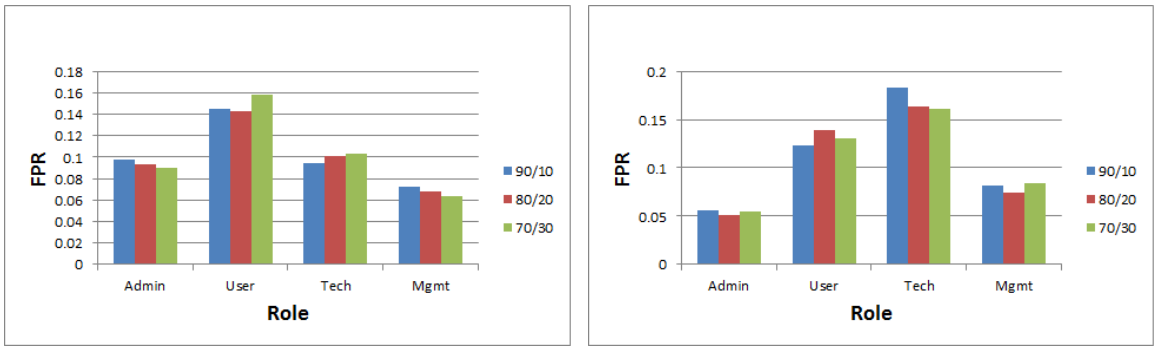
(a) Overall Role-based F-measure



(b) Overall User-based F-measure

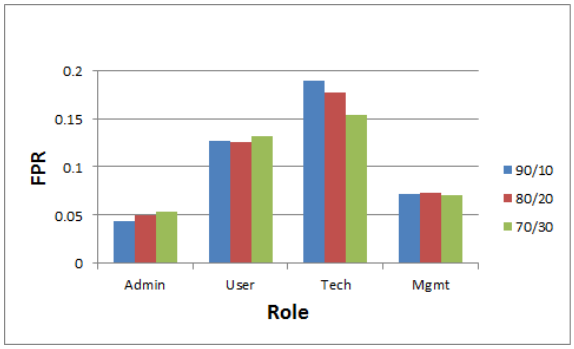
Figure 4.12: Overall F-measure

carefully analyzing the data set, we find that User1 has more sessions of activity (6 sessions) than User7 (2 sessions). Even though, User7 passed the initial criteria of

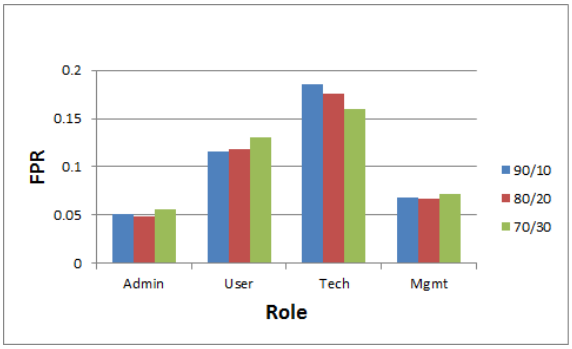


(a) N=2

(b) N=3



(c) N=4



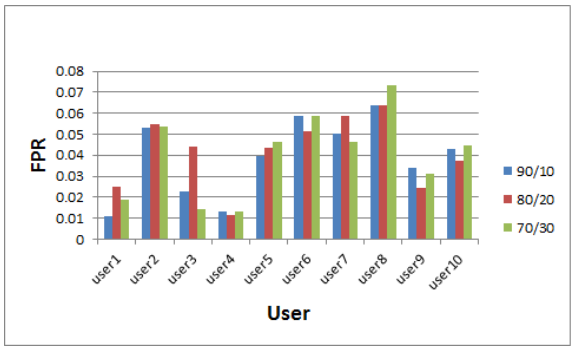
(d) N=5

Figure 4.13: Role-based False Positive Rate

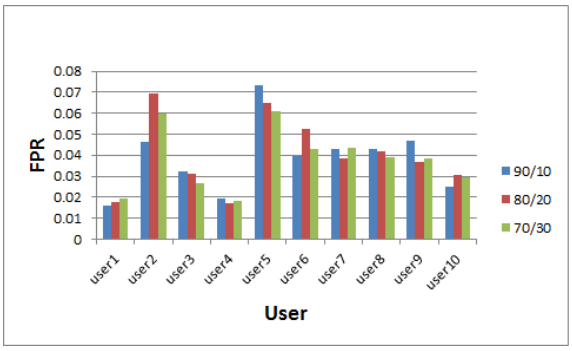
having at least 2 sessions of activity and 800 keywords, this user's f-measure is low. In this case, it is necessary to capture more session data per user.

The Admin and User roles have approximately the same number of keywords for training and their f-measure performance is approximately the same for data models N=3 through N=5 (see Figure 4.10)[1]. This shows that there could be some correlation in performance when classifying such models with the same range of keywords. To train individual users and category of users well, there is a noticeable difference in the number of keywords needed. For example, higher f-measure rates

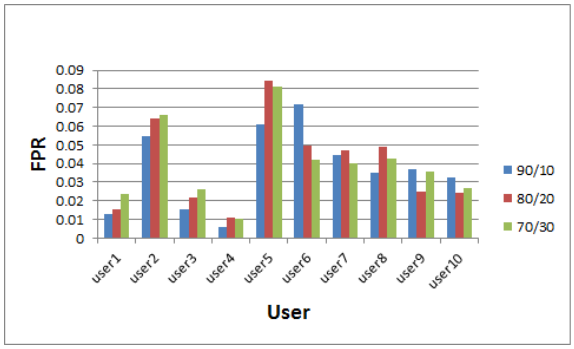
[1]The difference in the number of keywords for Admin and User is approximately 2000. This is the closest range for all categories observed in the dataset.



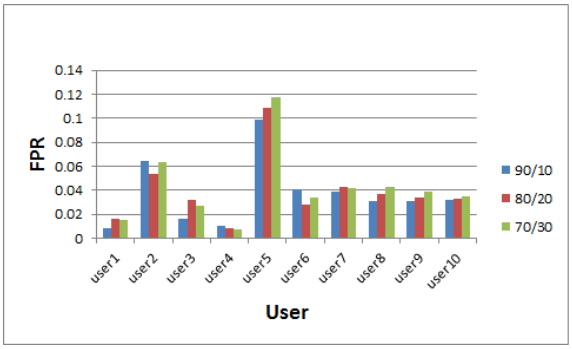
(a) N=2



(b) N=3

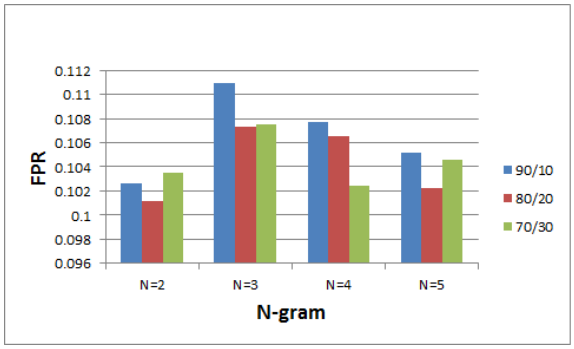


(c) N=4

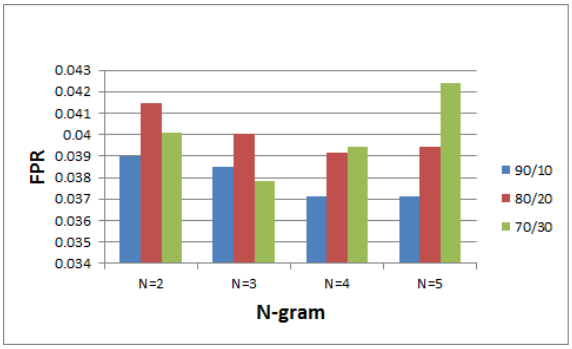


(d) N=5

Figure 4.14: User-based False Positive Rate



(a) Overall Role-based FPR



(b) Overall User-based FPR

Figure 4.15: Overall FPR

were received for users with approximately 850 keywords and 120 sessions (90/10 data split) and roles with approximately 30,000 keywords and 400 sessions (80/20

and 70/30 data split). The f-measure, on average, for role-based bi-grams perform at the highest and approximately the same rate (62%) for each data split (Figure 4.12-a). However, only the bi-gram 90/10 data split for user-based f-measure performs the highest at 60% (Figure 4.12-b).

The User role has the highest FPR for bi-grams using each data split with 70/30 data split having a slightly higher FPR (approximately 16%). However, when observing the classification of users with this role (i.e., User3), the 70/30 data split performs best at approximately 1% FPR. As N increases to three, four, and five, the Tech role has the highest FPR for each data split with the 90/10 data split having the highest/worst FPR for each model. It is important to emphasize that this data set has only 14% of keyword data labeled as User and 59% labeled as Tech. Since the Tech role is more prevalent than all roles combined, we observed an increased FPR for this role. Overall, we obtain less than 5% FPR for User-based models with $N = 5$, 90/10 data split models having the least FPR (Figure 4.15-b).

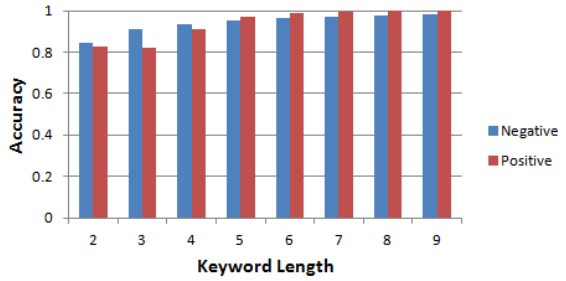
4.1.2.2 Binary Classification

When addressing RQ3, we want to consider whether the role-specific profiles constructed as part of RQ1 are capable of detecting outliers in user behavior. To evaluate this research question, we use the models generated from RQ1 independently in a binary classification approach, as described in Section 3.3.3. Because this task offers only two possible classes, a random model with no observation would achieve 50% accuracy in categorizing user activity as PASS or FAIL according to

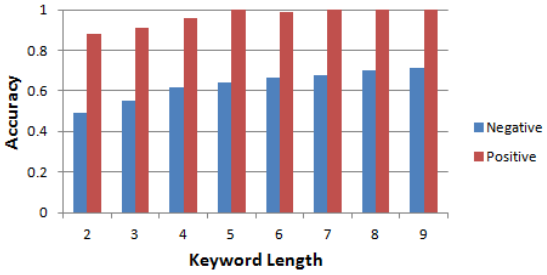
each role-specific model. As in RQ1, we expect **ID** to outperform this random baseline.

This task uses models independently. Specificity (i.e., *1-FPR*) and sensitivity (i.e., recall) are statistical measures used for the performance of this classification task. Both training and test data from the remaining three roles are used to evaluate the model's ability to reject uncharacteristic sequences (i.e., negative examples) by measuring specificity. Only test data from the model's own role is used to evaluate its ability to accept valid sequences (i.e., positive examples) using sensitivity measures. Additionally, we consider the affect of test sequence length on the performance of this task by evaluating the number of keywords of an input sequence. Therefore, even more test data for this task is obtained by separating user sessions into subsequences of keywords of length two to length nine.

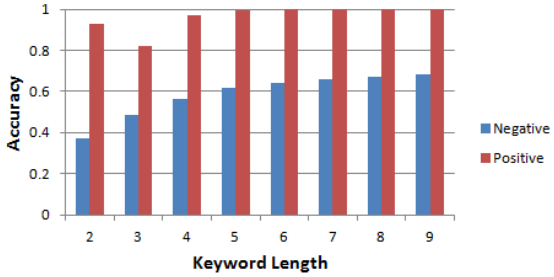
Finally, because we would like to track only a single threshold value for this task even though sequence length varies, there is a compensation for the fact that probability naturally tends to decrease as sequence length increases. In other words, a threshold approach which might perform well for input sequences of length two would likely overestimate the threshold for longer lengths. As an alternative, we adapt the model's output probability to be an indicator of entropy, a measure of uncertainty in a system. This provides the ability to normalize by the length of the input sequence. By doing so, a single threshold value for the binary classification task is maintained. Binary classification results are show in Figure 4.16.



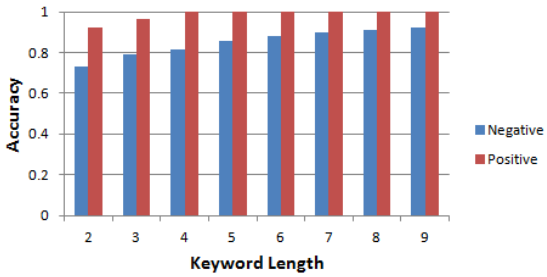
(a) User n-gram model



(b) Admin n-gram model.



(c) Technologist n-gram model.



(d) Management n-gram model.

Figure 4.16: Role-based Binary Classification

RQ3 relates most directly to the CUA goal of **ID**. Efficient experimental results are observed when accepting or rejecting snippets of user sessions based on role. Each model was tested against every available subsequence of user keywords, from lengths two to nine with a probability threshold of -0.6. Best performance was observed when using N=9. Recall that backoff and smoothing in the model allow for the assignment of probabilities to a sequence of any length, regardless of the maximum history considered. Note that accuracy is plotted separately for positive and negative test samples to analyze the models ability to reject uncharacteristic sequences (i.e., specificity) and accept valid sequences (i.e., sensitivity).

As expected, the length of sessions provided to models significantly affects the ability to correctly classify the example. The effect of length was much greater on negative examples, as failures due to rejecting a positive sample are rare after lengths greater than four. The User model has the most sessions of activity and performs at a level greater than 90% on all samples for lengths greater than three. When rejecting uncharacteristic sequences, the Management model eventually achieved performance of 92%, though this took sessions of length nine. With a 9% prevalence rate, it is evident that this role needs more training/test data for the identification of outliers.

4.2 Discussion

It's important to identify the prevalence of labeled data by determining how often a particular label occurs in the dataset. The dataset under evaluation is heavily populated with Tech-level keywords (approximately 59%). Therefore, the weighted harmonic mean of precision and recall (i.e., f-measure) is highest for this particular role at approximately 83% for each data split. For the multi-class approach, bi-grams seem to have the most stability but its evident that more training data is needed to reduce the false positive rate. We also conclude that individual user identification requires less keywords and per-session data than role identification.

The parameters influencing the model grow exponentially with N and hence a 5-gram model may not be practical in all cases. Having a larger value for N may also require a larger training set since patterns tend to be more sparsely distributed across keyword sessions. However, the parameters influencing a model, based on

web logs, with a larger N value can be reduced by dropping high-frequency 'noise' such as Joint Expert Group (.jpg), Cascading Style Sheets (.css), JavaScript (.js), Extensible Markup Language (.xml), and system generated text. Reducing this 'noise' gave us the ability to use a larger N value for the binary classification task. However, we saw very little improvement for increased values of N when roles are considered. From our results, one can conclude that binary classification proves to be much more effective than a random baseline at detecting uncharacteristic user behavior. Due to a large data set and elevated level of privileges for tasks, it was expected that the Tech user role, for binary classification, would have one of the strongest models but this was not observed. With the multi-class classification approach, we observed below optimal classification rates for the User role model. Conversely, the User model, in the binary approach, is stronger. This improvement in performance could be due to the use of shorter sessions which are less likely to contain unseen events. In this case, we rely on the validity of smoothing assumptions for accurate probability estimation.

Throughout this study, it has been evident that many advantages and disadvantages exist for statistical language models. These machine learning models contain flexible training and include test data that can be used to update execution strategies. If appropriate training is available, our models are capable of detecting malicious or unintended usage. In contrast, if relevant training is not available, poor prediction performance is unavoidable. This approach is also highly dependent on assumptions made by a system administrator. Therefore, detection performance can be affected by slight changes in the keyword abstraction process.

4.3 Threats to Validity

Intruder Detector is a user authentication tool that makes effective use of web logs generated in a real-world setting. Keywords are captured over a period of three years. Therefore, we train our n-gram classification models on realistic user behavior. We attempt to limit the risk of over-fitting the model to specific sequences of training data by selecting a diverse set of training examples for each test run. However, threats to external, internal and construct validity still exist.

Threats to external validity. Threats to external validity are factors that inhibit or reduce our ability to generalize our results. We validate our approach on *one* web-based application, for which user behavior is abstracted into a model and used to develop training and test sequences. Initially, the model was refined manually, to identify keywords that are specific to user behavior (e.g., .jsp entries) for this specific application. Some discrepancies may exist due to this action. If used in a real-world setting, we expect system administrators to know enough about their application to define instances in web log files that are indicative of user behavior. However, we expect similar results for applications in this domain. We must note that generalization beyond web-based organizational information system applications require additional experimentation and are not related to the results of this research study.

Threats to internal validity. Threats to internal validity occur when alternative causes for experimental results are present. The dataset used has a large class imbalance that may lead to an *accuracy paradox* for classification-based systems. We

introduce several metrics to assess the performance of our approach to remedy this concern. Since we use a real-world system with approximately 4000 registered users, we were unable to provide active users with a list of tasks to perform. Therefore, we depend on users to maintain consistency with their online behavior.

Threats to construct validity. Threats to construct validity are concerned with the lack of compatibility or similarity between the concept measures used in the study and the concepts intended to be measured. We use various performance assessment techniques to understand the effectiveness of our approach. However, metrics used to evaluate user behavior in this study may not be useful metrics in other domains using the same CUA technique.

4.4 Summary

In this chapter, we presented an empirical evaluation of CUA for web-based organizational information system applications. We answered three research questions with a custom set of evaluation metrics. Below is a high-level summary of our results:

- **RQ1**: *Role-based identification.* Roles with the best f-measure performed at 82% and approximately 10% FPR with bi-gram models. Overall, models with more per-session data have higher precision rates and models with more keyword data have higher recall rates. Training data plays a huge role in obtaining an accurate result.

- **RQ2**: *User-based identification.* The role of each individual user has a signif-

icant impact on user identification. Users that generated the most keywords have the best f-measure at 88% and approximately 10% FPR. Overall, the FPR is lower for these models when compared to role-based models.

- **RQ3**: *Identification of outliers.* From these experiments, binary classification has the most promising results. Overall, we obtain above 90% accuracy when accepting positive data and rejecting uncharacteristic data. Models with a larger set of keyword data reject uncharacteristic data at a lower rate than models with smaller sets of keyword data. Additionally, models that have more training sessions reject negative data at a rate greater than 80%.

Chapter 5: Conclusions

In this chapter, we present an overview of our research and discuss various opportunities for future work.

5.1 Research Overview

Many organizations are becoming increasingly aware of their security posture. It's important to identify cybersecurity-related risks and develop strategies to mitigate them. Continuous user authentication is one approach that can be utilized to identify impersonations and misappropriation of authentication credentials [61]. Specifically, this technique can be used with web applications to provide reliable and secure authentication. In this work, we present challenges that occur when modeling the behavior of users that interact with web-based software. We then present an approach to address the need for web-based continuous user authentication.

We obtain less than optimal results when conducting a feasibility study, using a real-world application, to assess the need for our approach. After exploring various statistical language models, we employ the use of n-grams to capture user interaction with web-based software. We use n-grams to model sequences and subsequences of user actions, their orderings, and the temporal relationships that make

them unique. After learning the behavior, we illustrate the ability to classify the models using multi-class classification to identify role and/or individual user characteristics. Results show model-specific differences in user behavior with performance highly dependent on session and keyword size. We identify outliers in variable-length keyword sequences using the binary classification technique. Results from this approach show the rate at which each model rejects uncharacteristic sequences and accepts valid sequences. Our CUA implementation is continual, non-intrusive, and behavioral.

In summary, the contributions of this research include the following:

- We develop a novel keyword abstraction technique to pre-process large volumes of web logs. This method helps reduce incomplete, noisy, and inconsistent data.

- We explore various statistical language models to capture the behavior of users as they interact with web-based organizational information systems.

- We develop a continuous user authentication framework, based on n-grams, to classify user sessions into roles and individual usage profiles. We also use this approach to identify outliers in role-based user behavior.

- We introduce a set of evaluation metrics to test the feasibility of our approach.

5.2 Future Work

Our work in continuous user authentication is a foundational approach for web-based user behavioral analysis. There are various ways our work can be extended to explore this research domain. We conclude that with two of the four role-based models considered, the correct identification of negative samples was above 90% as well as a 100% correct acceptance of positive samples. These findings are promising, and motivate future work to better understand model-specific differences which make this task more difficult for some cases (e.g., Admin and Technologist roles) than others. In particular, the finding that User sessions can so easily be protected against uncharacteristic usage is promising.

Many web applications have different sensitivity levels for security threats. Some systems must be protected at higher levels than other systems. There is a need to investigate an adaptive scheme (i.e., tunable parameter) to determine thresholds for different applications. The thresholds may be adjusted based on a user's role, application infrastructure, business logic, usage patterns, etc.

In many settings, statistical language models are computationally intensive. Building n-grams over large datasets pose challenges to memory and speed [62]. The computational cost of running these models should be calculated in real time to access the current usability performance. There may be a need to utilize a high performance computing environment to increase prediction time in an effort to help prevent or interrupt malicious web use. It will be useful to explore parallelization on various platforms such as shared memory processor machines and Linux clusters with

a high-speed network. Instead of starting with a parallel programming framework such as MPI or OpenMP, many standard tools exist. Parallelization scripts from the IRST Language Modeling Toolkit are an acceptable starting point to distribute tasks to a cluster of machines [63].

There are various performance metrics available for classification tasks. The performance metrics used in this work can be extended to include the time it takes to identify legitimate and malicious use. When outliers are identified, it's important to detect this activity before the active session is complete. However, in some cases, this time metric may allow some flexibility to help application forensics analyst gather reliable evidence to use against perpetrators.

A combination of machine learning techniques for classification should be explored for continuous user authentication. In this research, we use supervised learning techniques to access the classification of user behavior. Exploring how semi-supervised and unsupervised learning help make data-driven predictions or decisions is an open area of research in this domain.

Finally, our work uses a single modality for CUA tasks. The use of a multi-modal approach may prove to provide a more transparent authentication process and help reduce false positive rates. Even though our models are able to detect legitimate users and outliers, the false alarms can be annoying and decrease usability dramatically. For example, a combination of web logs, database logs, and graphical user interface (GUI) accesses can be used to obtain more information about a user's behavior. Keyboard and mouse interactions may also be integrated to provide a cost-effective solution for continuous authentication. At any time, when one modality

is not available, another one is able to capture such behavior and provide a highly predictive system.

In summary, in this dissertation, we have shown how sequences and subsequences of user interactions are captured within statistical language models to continuously identify users, user types, and outliers in user behavior. To provide focus, we have limited our research to web-based organizational information systems. However, the CUA domain can be extended to context-aware computing, Internet of Things (IoT), and people-centric applications.

Bibliography

[1] Richard P. Guidorizzi. Security: Active authentication. *IT Professional*, 15(4):4–7, 2013.

[2] SplashData. Splashdata news, 2015. [Online; accessed 03-October-2015].

[3] Koichiro Niinuma, Anil K. Jain, Jain B.V.K.V Kumar, S. Prabhakar, and A. A. Ross. Continuous user authentication using temporal information. *SPIE.*, 7667, 2010.

[4] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, 51(12):3448–3470, August 2007.

[5] Koichiro Niinuma, Unsang Park, and Anil K. Jain. Soft biometric traits for continuous user authentication. *Trans. Info. For. Sec.*, 5(4):771–780, December 2010.

[6] DARPA. Active authentication: http://www.darpa.mil/program/active-authentication, 2013. [Online; accessed 24-November-2015].

[7] Chao Shen, Zhongmin Cai, and Xiaohong Guan. Continuous authentication for mouse dynamics: A pattern-growth approach. In *Proceedings of the 2012 42Nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, DSN '12, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society.

[8] Fabian Monrose and Aviel D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Gener. Comput. Syst.*, 16(4):351–359, February 2000.

[9] Information System. Information system, 2015. [Online; accessed 03-October-2015].

[10] Leslie Milton, Bryan Robbins, and Atif Memon. N-gram based user behavioral model for continuous user authentication. In *The Proceedings of the Eighth International Conference on Emerging Security Information, Systems, and Technologies (SECURWARE 2014)*, 2014.

[11] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall, 2nd edition, 2009.

[12] Yubico. Yubico. http://www.yubico.com/, 2013. [Online; accessed 1-November-2013].

[13] Dino Schweitzer, Jeff Boleng, Colin Hughes, and Louis Murphy. Visualizing keyboard pattern passwords. *Information Visualization*, 10(2):127–133, April 2011.

[14] Andrew J. Klosterman and Gregory R. Ganger. Secure continuous biometric-enhanced authentication. Technical report, Carnegie Mellon Univ., May 2000.

[15] Michael Kaminsky, George Savvides, David Mazieres, and M. Frans Kaashoek. Decentralized user authentication in a global file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 60–73, New York, NY, USA, 2003. ACM.

[16] Richard Chow, Markus Jakobsson, Ryusuke Masuoka, Jesus Molina, Yuan Niu, Elaine Shi, and Zhexuan Song. Authentication in the clouds: A framework and its application to mobile users. In *Proceedings of the 2010 ACM Workshop on Cloud Computing Security Workshop*, CCSW '10, pages 1–6, New York, NY, USA, 2010. ACM.

[17] The Wall Street Journal. U.s. suspects hackers in china breached about 4 million people's records, officials say: http://www.wsj.com/articles/u-s-suspects-hackers-in-china-behind-government-data-breach-sources-say-1433451888, 2015. [Online; accessed 17-November-2015].

[18] Eric Grosse and Mayank Upadhyay. Authentication at scale. *IEEE Security and Privacy*, 11:15–22, 2013.

[19] I. Deutschmann, P. Nordstrom, and L. Nilsson. Continuous authentication using behavioral biometrics. *IT Professional*, 15(4):12–15, July 2013.

[20] Harini Jagadeesan and Michael S. Hsiao. Continuous authentication in computers. *Continuous Authentication using Biometrics: Data, Models, and Metrics*, 1:40–66, 2012.

[21] Alphan Altinok and Matthew Turk. Temporal integration for continuous multimodal biometrics. In *In Multimodal User Authentication*, pages 131–137, 2003.

[22] Hang-Bong Kang and Myung-Ho Ju. Multi-modal feature integration for secure authentication. In *Proceedings of the 2006 International Conference on Intelligent Computing - Volume Part I*, ICIC'06, pages 1191–1200, Berlin, Heidelberg, 2006. Springer-Verlag.

[23] Yinglian Xie, Fang Yu, Qifa Ke, Martin Abadi, Eliot Gillum, Krish Vitaldevaria, Jason Walter, Junxian Huang, and Zhuoqing Morley Mao. Innocent by association: Early recognition of legitimate users. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 353–364, New York, NY, USA, 2012. ACM.

[24] Pang Wu, Joy Zhang, Jiang Zhu, and Xiao Wang. Sensec: Mobile security through passive sensing. In *Proceedings of the 2013 International Conference on Computing, Networking and Communications (ICNC)*, ICNC '13, pages 1128–1133, Washington, DC, USA, 2013. IEEE Computer Society.

[25] Hataichanok Saevanee, Nathan Clarke, Steven Furnell, and Valerio Biscione. Continuous user authentication using multi-modal biometrics. *Computers and Security*, 53:234 – 246, 2015.

[26] Tee Kiah Chia, Khe Chai Sim, Haizhou Li, and Hwee Tou Ng. Statistical lattice-based spoken document retrieval. *ACM Trans. Inf. Syst.*, 28(1):2:1–2:30, January 2010.

[27] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM.

[28] Deyi Xiong, Min Zhang, and Haizhou Li. Enhancing language models in statistical machine translation with backward n-grams and mutual information triggers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1288–1297, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[29] Michael Collins, Brian Roark, and Murat Saraclar. Discriminative syntactic language modeling for speech recognition. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 507–514, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[30] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE*, page 2000, 2000.

[31] M. Egmont-Petersen, D. de Ridder, and H. Handels. Image processing with neural networksa review. *Pattern Recognition*, 35(10):2279 – 2301, 2002.

[32] Adam Fadlalla and Chien-Hua Lin. An analysis of the applications of neural networks in finance. *Interfaces*, 31(4):112–122, July 2001.

[33] MariaGraa Ruano and AntnioE. Ruano. On the use of artificial neural networks for biomedical applications. In Valentina Emilia Balas, Jnos Fodor, Annamria R. Vrkonyi-Kczy, Joszef Dombi, and Lakhmi C. Jain, editors, *Soft Computing Applications*, volume 195 of *Advances in Intelligent Systems and Computing*, pages 433–451. Springer Berlin Heidelberg, 2013.

[34] H Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *J. Machine Learning Research*, 10:1–40, 2009.

[35] Rajendra Akerkar and Priti Sajja. *Knowledge-Based Systems*. Jones and Bartlett Publishers, Inc., USA, 1st edition, 2009.

[36] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký. Strategies for training large scale neural network language models. In *Proceedings of ASRU 2011*, pages 196–201. IEEE Signal Processing Society, 2011.

[37] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Comput. Linguist.*, 22(1):39–71, March 1996.

[38] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language*, 10:187–228, 1996.

[39] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza, and Robert L. Mercer. Readings in speech recognition. chapter A Tree-based Statistical Language Model for Natural Language Speech Recognition, pages 507–514. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[40] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[41] Leonard A. Breslow and David W. Aha. Simplifying decision trees: A survey. *The Knowledge Engineering Review*, 12:1–40, 1 1997.

[42] Mikhail Petrovskiy. A data mining approach to learning probabilistic user behavior models from database access log. In Joaquim Filipe, Boris Shishkov, and Markus Helfert, editors, *Software and Data Technologies*, volume 10 of *Communications in Computer and Information Science*, pages 323–332. Springer Berlin Heidelberg, 2008.

[43] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

[44] Mukund Deshpande and George Karypis. Selective markov models for predicting web page accesses. *ACM Trans. Internet Technol.*, 4(2):163–184, May 2004.

[45] Eren Manavoglu, Dmitry Pavlov, and C. Lee Giles. Probabilistic user behavior models. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 203–, Washington, DC, USA, 2003. IEEE Computer Society.

[46] Leonard E. Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966.

[47] Leonard E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bull. Amer. Math. Soc.*, 73(3):360–363, 05 1967.

[48] Leonard E. Baum, T. Petrie, G. Soules, and N. Weiss. Maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(6):360–363, 12 1970.

[49] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[50] L. Razmerita. An ontology-based framework for modeling user behavior;a case study in knowledge management. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(4):772–783, July 2011.

[51] CAC. Dod common access card: http://www.cac.mill/, 2015. [Online; accessed 11-September-2015].

[52] James V. Wertsch. *Mind as Action*. Oxford University Press, 1998.

[53] Kenneth Burke. *Language as Symbolic Action*. University of California Press, 1966.

[54] Jimmy Lin and W. John Wilbur. Modeling actions of pubmed users with n-gram language models. *Inf. Retr.*, 12(4):487–503, August 2009.

[55] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1:5–32, 1999.

[56] Yiguang Liu, Zhisheng You, and Liping Cao. A novel and quick svm-based multi-class classifier. *Pattern Recogn.*, 39(11):2258–2264, November 2006.

[57] Paul Honeine, Zineb Noumir, and Cdric Richard. Multiclass classification machines with the complexity of a single binary classifier. *Signal Processing*, 93(5):1013 – 1026, 2013.

[58] Adam Pauls and Dan Klein. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 258–267, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[59] Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, 1996.

[60] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 181–184 vol.1, 1995.

[61] Issa. Traore and Ahmed A.E. Ahmed. Introduction to continuous authentication. In *IT Policy and Ethics: Concepts, Methodologies, Tools, and Applications*.

[62] Lars Bungum and Bjorn Gamback. Efficient n-gram language modeling for billion word web corpora. In *Workshop on Challenges in the Management of Large Corpora*, LREC '12, 2012.

[63] M. Federico, N. Bertoldi, and M. Cettolo. Irstlm: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech*, 2008.