User Identification Through Command History Analysis

Foaad Khosmood
Department of Computer Science
California Polytechnic State University
San Luis Obispo, CA 93407
foaad@calpoly.edu

Phillip L. Nico
Department of Computer Science
California Polytechnic State University
San Luis Obispo, CA 93407
pnico@calpoly.edu

Jonathan Woolery
Department of Computer Science
California Polytechnic State University
San Luis Obispo, CA 93407
woolery@calpoly.edu

***Abstract*—As any veteran of the editor wars can attest, Unix users can be fiercely and irrationally attached to the commands they use and the manner in which they use them.**

In this work, we investigate the problem of identifying users out of a large set of candidates (25–97) through their command-line histories. Using standard algorithms and feature sets inspired by natural language authorship attribution literature, we demonstrate conclusively that individual users can be identified with a high degree of accuracy through their command-line behavior. Further, we report on the best performing feature combinations, from the many thousands that are possible, both in terms of accuracy and generality.

We validate our work by experimenting on three user corpora comprising data gathered over three decades at three distinct locations. These are the Greenberg user profile corpus (168 users), Schonlau masquerading corpus (50 users) and Cal Poly command history corpus (97 users). The first two are well known corpora published in 1991 and 2001 respectively. The last is developed by the authors in a year-long study in 2014 and represents the most recent corpus of its kind. For a 50 user configuration, we find feature sets that can successfully identify users with over 90% accuracy on the Cal Poly, Greenberg and one variant of the Schonlau corpus, and over 87% on the other Schonlau variant.

I. Introduction

A Unix users command line behavior can reveal information about their identity. Researchers have taken advantage of this phenomenon to the problem domains of user profiling [1], intrusion detection [2][3][4], and intelligent digital assistance [5][6][7]. Of course user behaviors are subject to the affordances of the underlying systems. A two-button interface, for example, will not have much informative data distinguishing the behavior of its multiple users.

Previous work in intrusion detection has shown success in detecting when an attacker is masquerading as a known user by noticing when the users behavior deviates from his or her previous profile [8]. In the not-uncommon case when the attacker is an insider, it is valuable to also know the identity of the masquerader. Natural Language Processing (NLP) methods have shown great success in authorship attribution in documents when the pool of possible authors is known [9]. In contrast to natural languages, however, the language of Unix shell commands is a highly constrained and artificial one. In this work, we apply proven NLP methods to the problem of determining the identity of an inside attacker.

The Unix shell command-line offers a rich menu of choices to the user and multiple converging paths for the same task. The intruder identification problem can be framed as a supervised machine learning exercise whereby a record of Unix shell command usage by an unknown user can be analyzed and statistically correlated to a corpus of known users. Exercises of this kind have been carried out before, but the existing literature is focused on masquerading or classification based on general profiles such as novice or expert programmers, rather than individual users. The authorship attribution literature, which does often have the goal of isolating a single author, has traditionally focused on natural language corpora and relatively small candidate sets.

In this paper, we explore user attribution with large candidate sets (25 to 97 users), using machine learning techniques. We frame the specific problem as such: Given a set of user behavior profiles (command line histories) belonging to N known authors that we call candidates C_1 to C_N, can we attribute a previously unexamined command history portion of size S lines to one of the candidates with a high degree of accuracy?

In section II of this paper, we discuss the background in several related disciplines. In section III we discuss the three corpora that we use in our experiments. In section IV, we cover our methodology including algorithms and features. In section V, our experiments and results are discussed and finally in section VI, we provide conclusions and future work.

II. Background and related work

Source attribution, or the process of identifying the originator of a given textual work, has been used by researchers in several distinct disciplines. As far back as 1439, stylometry was employed by Lorenzo Valla to argue that the Donation of Constantine—the Roman Imperial Decree by which Constantine I famously transferred authority over Rome and the western part of the Roman Empire to the Pope—was in fact a forgery [10]. Historically source attribution finds its roots in linguistics and corpus studies as authorship attribution. The traditional approach to authorship detection uses stylometric analysis, and is generally performed manually [11]. Stylometry relies on the habitual and idiosyncratic application of linguistic style—that is: syntactic, semantic, orthographic, and lexicographic patterns—to differentiate between unique authors. With the advent of modern day computing however,

978-1-4799-4521-4/14/$31.00 ©2014 IEEE

computational methods have been applied to the tasks of user profiling [12][1][11], generation of adaptive user environments [5][7], user command prediction [6][13][7][14], and intrusion/anomaly detection [2][3][15][16][17].

Perhaps the most widely recognised modern example of authorship attribution is Forsyth and Holmes's seminal study of the Federalist Papers. Where the earlier work of Mosteller and Wallace started the exploration of non-traditional authorship attribution methods [18], Forsyth and Holmes broke new ground by employing bigrams and character n-grams to argue the authorship of these texts [19]. In more recent research, Stamatatos and his associates have had success in authorship attribution on literature using n-gram analysis [9][20][21]. The authors in [21] actually evaluated different parameters for their n-gram analysis and determined character trigrams to be particularly informative across several dimensions.

Similar methods have been employed with the goal of intrusion detection. These methods analyze historically established norms of behavior for individual users [3][4]. Much of the effort to employ computers in the task of authorship attribution has been focused on literary and historical works [20]. Other scholars focus upon computer users themselves. Leveraging the attribution task on the command line, these researchers attempt to generate a user profile that can be used to automatically detect masqueraders via statistical analysis [1][16][4][17].

User profiling [11] is also employed towards the goal of command prediction [6][13][14] and the development of adaptive user interfaces [5][7]. In command prediction, one popular approach is to profile each user based on the frequency of their command use (first word unigrams) [6]. Korvemaker and Greiner took this further, attempting to predict full command lines using the Greenberg corpus [13]. Adaptive user interfaces can be used to change system behavior to suit user needs, infer user preferences, and predict user actions [5].

III. Data Sources and Privacy

In this work we use three corpora collected over a period of 25 years, allowing us not only to have a large number of candidate users, but also to observe how things may have changed over time.

The first body of command traces was collected by Saul Greenberg in 1988 [22]. This corpus comprises 168 users of varying levels of technical proficiency. Although otherwise anonymized, the users are divided into computer scientists, non-programmers, novice programmers, and experienced programmers. These traces contain not only the commands issued, but also all of the arguments passed to them.

The second body of command traces was collected in 2000 by Matthias Schonlau [8] as part of a study into detecting attempts by one user to masquerade as another. This corpus includes the traces of 50 anonymized users. Because these data were collected using the `acct` auditing mechanism, the traces include only command names, but not arguments, and certainly not typographical errors which might provide clues to identity.

The third body of command traces was collected at Cal Poly during the Winter and Spring terms of 2014. For these traces we recruit volunteers from an upper division computer science class, Introduction to Operating Systems. The class is taken by junior and senior computer science, software engineering and computer engineering students. All students have Unix accounts provided by the Computer Science department in their first year. This group would be considered experienced Unix users and programmers. The contributions were made near the end of the term and thus would likely include all the work required for the Operating Systems class including assignments that required the use of Unix accounts. 97 individuals contributed their shell history files to this corpus. These were almost exclusively bash shell histories.

TABLE I. Details of the experimental corpora

Corpus	Total Commands	Total Users	100-line Fragments	Ave. Frags. per user
Cal Poly	90 554	97	963	9.93
Greenberg	290 215	168	2820	16.79
Schonlau	726 900	50	719	14.38

TABLE II. Command line properties of the corpora

Corpus	Average Line Length (words)	Average Word Length (characters)
Cal Poly	2.47	5.99
Greenberg	1.88	4.19
Schonlau (Cropped)	1.00	4.46
Schonlau (Filtered)	1.00	4.49

A. User Privacy

Because user command histories record everything a user does there is the possibility of them capturing sensitive personal information. As with the Greenberg, and Schonlau corpora, we address this concern by anonymizing the user names. After collecting the traces, each username is randomly mapped onto one of the most popular names from the 1890 U.S. census, and all instances of that login name within the traces are replaced with the new alias. Access to the traces has been restricted to the analysis software without any human having looked at them.

The resulting experimental corpora are described in Table I. Some key features of the corpora are shown in Table II.

IV. Methodology

Major steps in our methodological workflow are: preprocessing, feature selection, algorithm selection, parameterization, training and finally evaluation. We derive a floating point accuracy value associated with each experiment which we use to compare the relative merits of features and parameters. Figure 1 shows the major steps in our workflow.

A. Preprocessing

We first create a profile per individual in the corpus. For the Cal Poly corpus (CP) where more than one history file could have been submitted per individual, we combine the files in order of chronology obtaining one large record per user. The Greenberg corpus (GB) is processed by extracting just the command line entries, discarding the labeled classes used in that study. The entries are combined into a single document per user.

The Schonlau data is preprocessed in two different ways. First we crop to the initial 5000 lines guaranteed to have

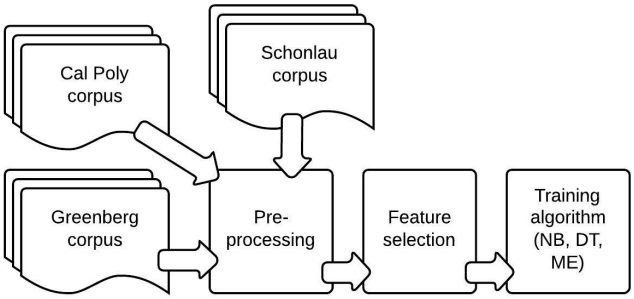

Fig. 1. Data processing workflow

no masquerading commands in them. That gives uniform traces of 5000 confirmed commands per user. This is the Schonlau-cropped corpus (SC) and is very much comparable to the other two corpora. In addition, we process the traces to remove the 100-line segments of unknown origin containing the masquerading commands. The result is a slightly different corpus we call Schonlau-filtered (SF) which yields longer traces, but of less uniform length.

Next, we break up each profile into one or more documents of 100 lines each. We discard the less than 100-line remainder of each profile, and we also dismiss the entire profile if it is less than 100 lines.

We do not take all of the standard preprocessing steps that computational linguists often recommend, as we believe our analysis would not benefit from them. These steps include: case collapsing, word-stemming, non-alphabetic character removal and punctuation removal. We do however, handle some unique shell features such as command delimiters and pipes when extracting features. A line appearing in a user's trace, for example, may contain a pipe symbol and thus would need to be processed as two commands with possibly two sets of flags and arguments.

B. Feature selection

The literature in areas of statistical authorship attribution, as well as machine learning for general text documents, contains many features found to be useful with different experiments. Stamatatos outlines the various properties of lexical, character, syntactic, semantic, and application-specific features in [9]. N-gram features, both at word and character level, are particularly popular, as are semantic features. Word n-grams and character n-grams have been used with some success in many studies [21]. In classification experiments success has been found using function words as features [23]. The most important criterion for selecting features in authorship attribution tasks is their frequency. In general, the more frequent a feature, the more stylistic variation it captures [9]. We divide features into these categories:

1) First Word (three types: distinct first words unconditional, first words that are known Unix bin commands, and first words that are known Unix sbin commands)
2) Limited semantic (two types of semantic features: Unix-style flags and numeric arguments)
3) Word n-grams (for n= 0, 1, 2, 3, and 5 words)

TABLE III. FEATURE SET LABELING USED IN FIGURES

Symbol	Feature
f	First word (command name)
s	Limited semantic labeling of words
wN	Word N-grams of length N
cN	Character N-grams of length N

4) Character n-grams (for n= 3, 6, and 9 characters)

Each feature in this analysis is binary. We use labels listed in Table III to refer to these feature categories throughout this paper.

C. Algorithms

Three fairly standard classification algorithms are used: Naïve Bayes (NB), Decision Trees (DT) and Maximum Entropy (ME). Implementations from the Natural Language Tool Kit (NLTK) [24] package are used. We discuss each of the methods below. We derive a floating point accuracy value associated with each classification experiment which we use to compare the relative merits of features and parameters.

The Naïve Bayes classifier is a probabilistic classifier. It makes the naïve assumption that the presence or absence of a given feature is unrelated to that of another feature. In text classification, the NB multinomial variant (MNB) remains a very popular method both for its simplicity and reliability [25][26]. MNB does often perform poorly compared to other models, and for this reason it is often used as a comparison baseline. For a more detailed discussion on strengths and weaknesses of MNB, see [27].

Although we use MNB, and thus consider prior probabilities based on total number of features observed, we do not count the features themselves more than once per document (binary features). Thus when calculating the overall probability, our implementation ignores the "0" occurrence cases, as a form of smoothing.

Decision Trees are logic based learning algorithms. The basic method is the construction of a tree where each node is a feature. During training, at each step, the algorithm searches through all the features and finds one that best divides the rest of the documents evenly in branches. The process is repeated at each branch until the leaves of the tree become individual classes. For a great overview see [28]. Our decision tree implementation is very similar to the ID3 algorithm [29].

The Maximum Entropy classifier, also known as multinomial logistic regression, works to increase the influence of those features that contribute most to documents being classified correctly during training. These classifiers iteratively distribute weights among all the terms of a linear equation to maximize the likelihood of correct classification. The resulting weight matrix is then applied to the features of the test set data to make a class prediction. An optimization function is required. We employ the conjugate gradient method to iteratively derive the best distribution. See [30] for more on the method.

V. EXPERIMENTS AND RESULTS

In order to determine the best performing feature combinations, we run a large number of classification experiments with

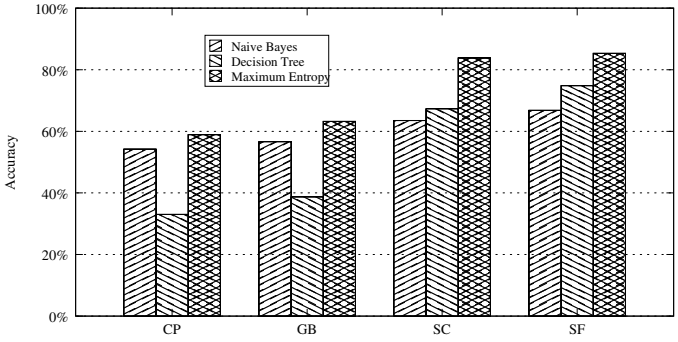

Fig. 2. All corpora, maximum size, using only first word

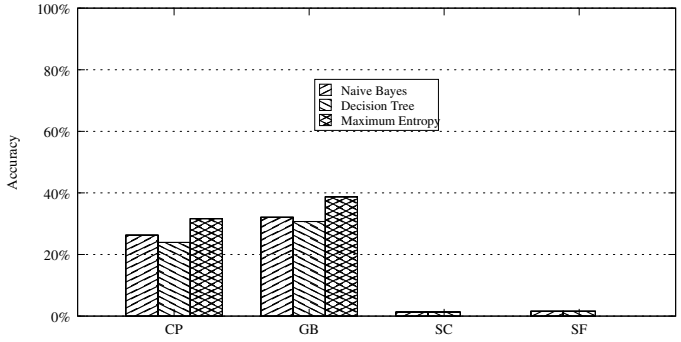

Fig. 3. All corpora, maximum size, using only "semantic"

different features, corpus sizes and algorithms. With respect to features in particular, there is no evidence that a superset of features performs as accurately or better than a smaller subset of them. In fact, frequently this is false due to increase of noise from bad features. Thus, we can only experimentally determine which subset performs best.

As outlined in the last section, we have four feature groups and any combination of them can be included in a classification experiment. When denoting a grouping or combination of features in the following tables, we use code letters possibly followed by integers to indicate the feature (see Table III). We experiment with all combinations of features. The first group is the "first word" features which we denote by "f". Another group, "s", detects numerals and Unix flags. Together there are four "fs" combinations including the case where neither is included. To these we add 5 variants of word n-grams "wng N" for N= 0, 1, 2, 3, 5; and four variants of character n-grams "cng N" with N= 0,3,6 and 9. Altogether, there are 80 feature combinations.

The feature groupings are applied to our four corpora with up to four different sizes[1], and three possible algorithms (NB, DT, ME). Thus we have a total of $80 \times 12 \times 3 = 2880$ unique classification experiments, each resulting in a floating point accuracy value.

When running each experiment with a certain size and corpus, we retrieve the records of the specified number of users of the given corpus randomly. We divide each user's trace into 100 line labeled documents. We then set aside one third of the total number of documents at random as our test set. We train with the other two thirds. Each experiment is run three times with a different random set of users and the accuracy results averaged.

We summarize our findings, first by analyzing the effect of individual features, then by attempting to evaluate their combined effect. In the interest of simplicity, in the following four figures, only the largest sample size for each corpus is considered since this represents the most difficult classification problem.

Figure 2 presents the results for all algorithms for all corpora using only the first word feature.

Figure 3 presents the results for all algorithms for all corpora using only the "semantic" feature group. Note that the Schonlau corpora contain only singular commands without any flags or argument, and therefore no features could be extracted from them for this experiment.

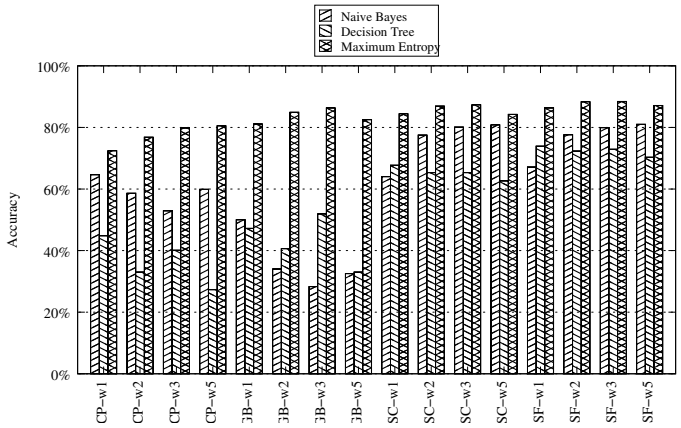

Fig. 4. All corpora, maximum size, using word n-grams

Figure 4 presents the results for all algorithms for all corpora using only the word n-grams of sizes 1, 2, 3, and 5.

Figure 5 presents the results for all algorithms for all corpora using only the character n-grams of sizes 3, 6, and 9.

Now let us examine the best performances. Table IV shows which feature sets and algorithms performs best for each corpus at each sample size. Feature sets are specified by the system as above. For example the feature set "fsw2c6" means the feature set contains all first-word features, semantic features, word bigram features and character 6-gram features.

Table V shows a list of the top 50 best performing feature combinations based on average performance over all four corpora and all 12 sample sizes. Every row is supposed to contain both a feature set and one of the three algorithms. However, every one of the top 50 best performing entries are by the ME algorithm, thus we do not denote the algorithm in the table. Many of the best performing features include both word and character n-grams and the absolute best is a simple word trigram. While several of the top 10 combinations include semantic features ("s"), there is always the same feature combination without the semantic features actually performing better, indicating the semantic features tend to hurt

[1]25 and 50 users for all four corpora; 70 and 97 users just for CP and GB, yields 12 corpus-size combinations in total.

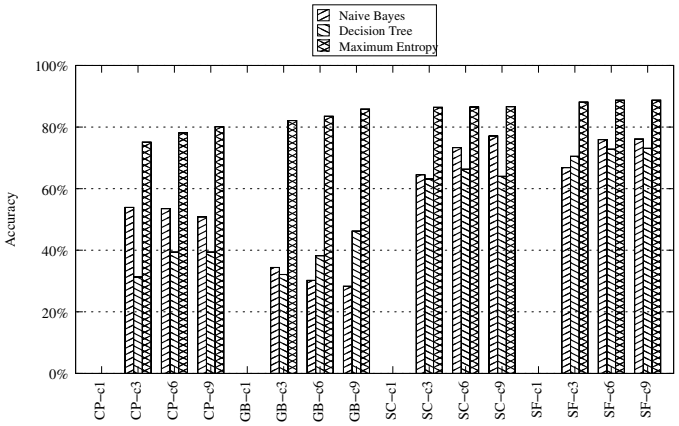

Fig. 5. All corpora, maximum size, using chracter n-grams

TABLE IV. BEST FEATURE SET AND ALGORITHM FOR EACH CORPUS AT EACH SIZE (ALGORITHM: MAX. ENTROPY IN ALL CASES)

Corpus	Size	Features	Accuracy
Cal Poly (CP)	25	w3	96.7%
	50	w5	93.0%
	70	w5	89.9%
	97	w5	80.5%
Greenberg (GB)	25	w2c6	93.1%
	50	w3c9	90.2%
	70	w2c9	87.1%
	97	w3	86.3%
Schonlau, cropped (SC)	25	fw1c9	93.6%
	50	fw5c3	88.6%
Schonlau, filtered (SF)	25	w5c3	92.5%
	50	w5c3	90.3%

performance, rather than help. Number 12 is the first that includes the first-word element ("f"). In general, the situation is very similar to semantic features. When either "f" or "s" appear in the set, it's typically a worse-performnig version of a better combination already listed ahead of it in ranking. It's clear that n-gram combinations, particuarly low number word n-grams and high number character n-grams are the most robust across all four corpora.

The overall best performing feature set / algorithm combination, averaging 89.1%, was "w3". Interestingly, the word trigram was also instrumental in achieving best results in [12]. Table VI shows the word trigram performance for each corpus at each size.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we show that Unix users can be individually identified through analysis of their command line usage with a high degree of accuracy. This is true even with almost 100 users as potential attackers. It is particularly useful in insider attack situations where the attacker may be using another user's credentials. For a 50 user corpus where we can experiment with all four of corpora, we are able to achieve accuracy rates of 87% to 93%. We further investigate which set of conditions leads to the best performance, experimenting with four different corpora with sizes ranging from 25 to 97, three different classification algorithms, as well as over 80 distinct feature combinations. We find that the ME algorithm, and word trigram features work best overall. More detailed observations are offered below.

TABLE V. TOP PERFORMING FEATURE SET/ALGORITHM PAIRS (ALGORITHM: MAX. ENTROPY IN ALL CASES)

		Accuracy				
Rank	Features	CP	GB	SC	SF	Overall
1)	w3	89.3	88.7	89.5	89.4	89.1%
2)	sw3	88.4	87.9	89.5	89.4	88.6%
3)	w3c9	86.6	89.9	89.1	90.0	88.6%
4)	w2c9	86.1	89.9	89.3	90.1	88.5%
5)	sw2c9	86.2	88.9	89.3	90.1	88.3%
6)	sw3c9	86.4	88.7	89.1	90.0	88.2%
7)	w5c9	86.6	88.3	89.4	90.1	88.2%
8)	c9	87.3	87.8	89.1	89.8	88.2%
9)	w5	90.0	86.5	86.6	88.3	88.0%
10)	sw5c9	86.4	87.2	89.4	90.1	87.8%
11)	w1c9	84.9	88.2	89.9	90.5	87.7%
12)	fw3c9	85.8	87.3	89.6	90.2	87.7%
13)	sc9	85.7	87.6	89.1	89.8	87.6%
14)	c6	85.6	87.0	89.0	89.8	87.5%
15)	fsw3c9	85.3	87.0	89.6	90.2	87.4%
16)	sw1c9	84.8	87.0	89.9	90.4	87.3%
17)	fc9	84.1	87.6	89.4	90.6	87.2%
18)	fw2c9	84.6	87.3	89.4	90.0	87.2%
19)	sw3c6	84.4	86.9	90.0	90.5	87.2%
20)	fsw2c9	84.7	87.0	89.4	90.0	87.1%
21)	fw5c9	85.4	85.8	89.6	90.6	87.1%
22)	sw2c6	83.4	88.2	89.3	89.9	87.1%
23)	fsw5c9	85.4	85.6	89.6	90.6	87.1%
24)	w2	84.3	87.3	88.9	89.7	87.0%
25)	w3c6	82.6	88.4	90.0	90.5	87.0%
26)	sc6	83.4	88.0	89.0	89.8	86.9%
27)	w2c6	82.9	88.7	89.3	89.9	86.9%
28)	fsc9	84.1	86.5	89.4	90.6	86.9%
29)	fsw3c6	83.8	86.2	90.1	90.5	86.8%
30)	fw1c9	83.6	86.0	90.6	90.6	86.8%
31)	w5c6	82.5	87.4	89.9	90.7	86.7%
32)	sw5c6	83.3	86.2	89.9	90.7	86.6%
33)	fw3c6	83.3	86.1	90.1	90.5	86.5%
34)	w3c3	82.2	86.8	90.3	91.0	86.5%
35)	fw5c6	82.8	85.8	90.3	91.0	86.4%
36)	sw2	83.7	86.2	88.9	89.7	86.4%
37)	fsw1c9	83.2	85.3	90.6	90.6	86.4%
38)	fw2c6	83.3	86.0	89.4	90.3	86.4%
39)	fsc6	83.0	86.7	89.1	89.7	86.4%
40)	w1c6	82.3	87.0	89.9	90.0	86.4%
41)	sw2c3	82.9	86.4	89.2	90.0	86.3%
42)	fsw5c6	82.6	85.7	90.3	91.0	86.3%
43)	fw2c3	82.5	86.7	89.8	89.8	86.3%
44)	fsw2c6	82.5	86.3	89.4	90.3	86.2%
45)	fw1c6	83.4	85.4	89.6	89.8	86.2%
46)	w2c3	81.6	87.2	89.2	89.9	86.0%
47)	fc6	83.0	85.7	89.1	89.7	86.0%
48)	sw1c6	82.1	86.1	89.9	90.0	86.0%
49)	fsw1c6	83.0	85.0	89.6	89.8	85.9%
50)	fsw2	82.2	85.3	89.9	90.2	85.9%

TABLE VI. WORD TRIGRAM (W3) FEATURE PERFORMANCE OVER ALL CORPORA AND SIZES

Corpus	Size	Accuracy
Cal Poly (CP)	25	96.7%
	50	91.4%
	70	89.2%
	97	79.8%
Greenberg (GB)	25	91.9%
	50	90.0%
	70	86.7%
	97	86.3%
Schonlau, cropped (SC)	25	91.7%
	50	87.3%
Schonlau, filtered (SF)	25	90.4%
	50	88.4%

A. Observations about feature sets

In our study, semantic and first-word features appear less consistently effective than other features. While in a few cases the "fs" feature combination provides higher accuracy, this feature set usually produces less accurate results than other features.

Word n-grams show themselves to be very strong stand-alone features, with word unigrams and bigrams providing many top results. Character n-grams were also shown to be exceptionally strong features. In addition to high accuracy, character n-grams provide the best examples of scalability, showing some of the highest results in our study while being run on the largest author sets. The 9-gram feature "c9" is the best overall character n-gram feature, appearing in many of our best results.

Although, we analyze many features and variations, more are possible and have been alluded to in the literature. More semantic features are possible: for example actual function category of commands like "editor", "graphics" or "software development tool". Extracting these features would require additional annotation. Given the low performance of the set of semantic features we already use, it would be interesting to see if more such features would improve accuracy.

B. Machine Learning approaches

Algorithms used are a good cross section of general methods, but more approaches are possible. Our three classifiers represent three different philosophies to classification. The important point is that none of the three classifiers are specialized and fine-tuned for this kind of data. They are considered generic text analysis methods in the NLP community. Inclusion of NB is almost a necessity in any machine learning comparison. NB is considered the classic generative model and often forms the baseline analysis for other algorithms. DT algorithms are logic based (as opposed to statistical). This kind of classifier has its own strengths and weaknesses. Decision tree classifiers are often oversensitive to small changes. The ME is the only discriminative model with constant weight redistribution. We suspect it was best performing because we had many features with uneven contributions. Both DT and ME have tendencies to over-fit the data. Our result indicates that ME classifiers are superior to either DT or NB for these experiments.

In addition to these, SVM and Boosting-based ensemble classifiers could be incorporated in future work.

C. Changes over time

There is an additional observation that can be made about the accuracy of user identification across our two most similar corpora, Greenberg and Cal Poly: The algorithms perform better on the older corpus given the same user sizes and features. We postulate this is because the nature of user interfaces has changed dramatically over time and command line usage has diminished[2]. As more applications move to GUIs and the web, less and less gets captured in command logs. The Greenberg corpus has 5365 unique commands, compared to Cal Poly, with only 1761 unique command. Individual user behavior is shown in Table VII. We did not include Schonlau's data here, because the command data were not taken from typed commands alone.

[2]Once upon a time, before the advent of the World Wide Web, even procrastination had to be command-line based: A quick check shows `rogue` and news readers in 33% of Greenberg's traces.

TABLE VII. UNIQUE COMMANDS PER USER

	Corpus	
	Cal Poly	**Greenberg**
Min.	16	11
Max.	184	394
Mean	59.5	91.8
S.D.	24.7	64.0

In conclusion, our user attribution experiments with three separate and distinct corpora allow us to identify some high performing feature combinations and algorithms. While many more combinations of features and algorithms are conceivable –and we have identified many for future work– the experiments we conduct result in high levels of accuracy for the given corpora, and allow us to consider important questions about the data and the process.

For a full list of experiments and results please visit http://www.csc.calpoly.edu/~foaad/UA_study.

ACKNOWLEDGMENT

The authors would like to thank the volunteers who were willing to share their command histories. Without their data, this work would not have been very interesting.

REFERENCES

[1] W.-H. Ju and Y. Vardi, "Profiling UNIX users and processes based on rarity of occurrence statistics with applications to computer intrusion detection," in *Proceedings of the Fourth International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, Davis, CA, 2001, pp. 1–10.

[2] S. Kumar, "Classification and detection of computer intrusions," Ph.D. dissertation, Purdue University, 1995.

[3] T. F. Lunt, "A survey of intrusion detection techniques," *Computers and Security*, vol. 12, no. 4, pp. 405–418, Jun. 1993.

[4] D. Geng, T. Odaka, J. Kuroiwa, and H. Ogura, "An n-gram and stf-idf model for masquerade detection in a unix environment," *Journal of Computer Virology and Hacking Techniques*, vol. 7, no. 2, pp. 133–142, 2011.

[5] A. Lee, "Investigations into history tools for user support," Ph.D. dissertation, University of Toronto, Toronto, Canada, 1992, tech. report CSRI-271.

[6] B. D. Davison and H. Hirsch, "Experiments in unix command prediction," in *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, ser. AAAI'97/IAAI'97. AAAI Press, 1997, pp. 827–827.

[7] S. Greenberg, J. Darragh, D. Maulsby, and I. H. Witten, "Predictive interfaces: What will they think of next?" Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, Tech. Rep. 1991-448-32, 1992.

[8] M. Schonlau and M. Theus, "Detecting masquerades in intrusion detection based on unpopular commands," *Information Processing Letters*, vol. 76, pp. 33–38, 2000.

[9] E. Stamatatos, "A survey of modern authorship attribution methods," *Journal of the American Society for Information Science and Technology*, vol. 60, no. 3, pp. 538–556, 2009.

[10] C. B. Coleman, *Discourse on the Forgery of the Alleged Donation of Constantine (Translation of: Valla, Lorenzo (1440). De Falso Credita et Ementita Constantini Donatione Declamatio)*. New Haven, CT: Yale University Press, 1922, hosted at the Hanover Historical Texts Project (http://history.hanover.edu/texts/vallatc.html).

[11] V. N. P. Dao, R. Vemuri, and S. J. Gempleton, "Profiling users in the UNIX OS environment," in *Proceedings of the International ICSC Conference on Intelligent Systems and Applications*, University of Wollongong, Australia, Dec. 2000.

[12] J. A. Iglesias, P. Angelov, A. Ledezma, and A. Sanchis, "Creating evolving user behavior profiles automatically," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 5, pp. 854–867, 2012.

[13] B. Korvemaker and R. Greiner, "Predicting unix command lines: Adjusting to user patterns," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press, 2000, pp. 230–235.

[14] B. D. Davison and H. Hirsh, "Predicting sequences of user actions," in *Proceedings of the 1998 AAAI/ICML Workshop "Predicting the Future: AI Approaches to Time-Series Analysis"*. AAAI Press, 1998, pp. 5–12.

[15] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, no. 3, pp. 151–180, Aug. 1998.

[16] R. A. Maxion, "Masquerade detection using enriched command lines," in *Proceedings of the International Conference on Dependable Systems and Networks*. San Francisco, CA: IEEE Computer Society, Jun. 2003, pp. 5–14.

[17] S. Shahidi, P. Mazrooei, N. N. Esfahani, and M. Saraee, "Proximity user identification using correlogram," *Intelligent Information Processing V (Proceedings of the 6th IFIP International Conference on Intelligent Information Processing)*, pp. 343–351, 2010.

[18] F. Mosteller and D. L. Wallace, *Inference and disputed authorship: The Federalist*. Addison-Wesley, 1964.

[19] D. I. Holmes and R. S. Forsyth, "The federalist revisited: New directions in authorship attribution," *Literary and Linguistic Computing*, vol. 10, no. 2, pp. 111–127, 1995.

[20] E. Stamatatos, N. Fakotakis, and G. Kokkinakis, "Automatic authorship attribution," in *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL 1999)*, 1999, pp. 158–164.

[21] J. Houvardas and E. Stamatatos, "N-gram feature selection for authorship identification," in *Proceedings of the 12th international conference on Artificial Intelligence: methodology, Systems, and Applications*, ser. AIMSA'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 77–86.

[22] S. Greenberg, "Using UNIX: collected traces of 168 users," Department of Computer Science, University of Calgary, Alberta, Canada, Tech. Rep. 88/333/45, includes tar-format cartridge tape, 1988.

[23] Y. Zhao and J. Zobel, "Effective and scalable authorship attribution using function words," in *Proceedings of the Second Asia conference on Asia Information Retrieval Technology*, ser. AIRS'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 174–189.

[24] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. O'reilly, 2009.

[25] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive bayes text classifiers," in *Proceedings of the Twentieth International Conference on Machine Learning*, Washington, DC, 2003, pp. 616–623.

[26] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752, 1998, pp. 41–48.

[27] E. Frank and R. R. Bouckaert, "Naive bayes for text classification with unbalanced classes," in *Knowledge Discovery in Databases: PKDD 2006*. Springer Berlin Heidelberg, 2006, pp. 503–510.

[28] S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data mining and knowledge discovery*, vol. 2, no. 4, pp. 345–389, 1998.

[29] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[30] J. R. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," 1994.