Removing Biases in Unsupervised Learning of Sequential Patterns

Yoav Horman and Gal A. Kaminka*
The MAVERICK Group
Department of Computer Science
Bar-Ilan University, Israel
{hormany,galk}@cs.biu.ac.il

February 12, 2008

Abstract

Unsupervised sequence learning is important to many applications. A learner is presented with unlabeled sequential data, and must discover sequential patterns that characterize the data. Popular approaches to such learning include (and often combine) frequency-based approaches and statistical analysis. However, the quality of results is often far from satisfactory. Though most previous investigations seek to address method-specific limitations, we instead focus on general (method-neutral) limitations in current approaches. This paper takes two key steps towards addressing such general quality-reducing flaws. First, we carry out an in-depth empirical comparison and analysis of popular sequence learning methods in terms of the *quality* of information produced, for several synthetic and real-world datasets, under controlled settings of noise. We find that both frequency-based and statistics-based approaches (i) suffer from common statistical biases based on the length of the sequences considered; (ii) are unable to correctly generalize the patterns discovered, thus flooding the results with multiple instances (with slight variations) of the same pattern. We additionally show empirically that the relative quality of different approaches changes based on the noise present in the data: Statistical approaches do better at high levels of noise, while frequency-based approaches do better at low levels of noise. As our second contribution, we develop methods for countering these common deficiencies. We show how to normalize rankings of candidate patterns such that the relative ranking of different-length patterns can be compared. We additionally show the use of clustering, based on sequence similarity, to group together instances of the same general pattern, and choose the most general pattern that covers all of these. The results show significant improvements in the quality of results *in all methods*, and across all noise settings.

Keywords: Sequence learning, sequential patterns, sequence mining

*This research was supported in part by BSF grant #2002401.

1 Introduction

Automated sequence learning is an important task in which a data learning system is presented with unlabeled sequential data, and must discover sequential patterns that characterize the data [3]. Applications include user modeling [4], anomaly detection [13], and system execution analysis [10, 11, 23].

Two popular approaches to this task are frequency-based (*support*) methods (e.g., [3,22]), and statistical dependence methods (e.g., [10]). These methods have been studied separately and in combination, often with the intent of the investigation on scalability and removal of spurious results. Yet the quality of sequential pattern learning results is still often far from satisfactory [11,18]. Much of recent literature is therefore devoted to examining method-specific flaws of various statistical and frequency-based methods (e.g., [1]). For instance, [5] discusses different flaws of several statistical methods often used in combination with support-based learning techniques.

In contrast, we seek to determine general (method-neutral) causes for the poor results of existing methods. In an effort to better understand the issues, we empirically compare the results produced by popular support-based and statistical sequential pattern learning methods on several synthetic and real-world data sets. The comparison uncovers several *general* deficiencies in *all* the methods we tested.

First, the results show that all tested methods are biased in preferring sequences based on their length, often preferring shorter sequences to more meaningful longer sequential patterns. Second, we find that all approaches are unable to correctly generalize the patterns discovered, thus flooding the results with multiple instances (with slight variations) of the same pattern. Finally, the results show that the relative quality of different approaches changes based on the noise present in the data: Statistical approaches do better at high levels of noise, while frequency-based approaches do better at low levels of noise.

We present methods for addressing all of these deficiencies. First, we present a *length normalization* method that leads to significant improvements in *all* sequence learning methods tested (up to 42% improvement in accuracy). We then show how to use clustering to group together similar sequences. We show that previously distinguished sub-patterns are now correctly identified as instances of the same general pattern, leading to *additional* significant accuracy improvements. The experiments show that the techniques are generic, in that they significantly improve all of the methods initially tested. All experiments are carried out in tens of thousands of repeated trials, on both synthetic and real-world data.

This paper is organized as follows. First, Section 2 presents a brief overview of the sequential pattern learning problem, and previous work, which has focused on method-specific limitations of various support-based and statistical data analysis approaches. Then (Section 3), we present the results from our empirical investigation of popular methods using synthetic and real-world datasets. The conclusions we draw from these results as to general biases in existing learning approaches are presented in Section 4, and this section also presents techniques for addressing these biases, regardless of the approach used. Section 5 presents the results of experiments with the techniques we develop, and demonstrates the significant improvements achieved. Section 6 concludes.

2 Background and Related Work

In unsupervised learning of sequential patterns, the learning system is given example streams, each a sequence $\alpha_1, \alpha_2, \ldots, \alpha_m$ of some atomic *events* (e.g., observed actions). The system must extract *sequential patterns*—sequences of events—which characterize the example streams, with as little assistance as possible. Of course, not every pattern is characteristic of the streams, as some of the patterns reflect no more than a random co-occurrence of events. Thus it is important to only extract patterns that are composed of correlated events and that reveal valuable information about the dataset. Different methods for sequence learning differ in their scalability (e.g., [16, 20, 22, 24]), in their goal settings (e.g., patterns allowing for arbitrary separation between events [3] or only for closely-occurring events [14]), and in how they characterize such patterns (i.e., in how they determine that a pattern is interesting). Our study in this paper focuses on determining limitations that are general, in that many different characterizations of interesting patterns suffer from them. We thus ignore here the issue of differences between methods based on scalability or event separation.

2.1 Frequency based methods

Perhaps the most common characterization of patterns is based on their relative frequency—called *support*—within the database [3]. The algorithm first finds all frequent sequences of size 1, i.e. all attributes whose support exceeds a user-defined threshold called *min_support*. Since the subsequences of each frequent 2-sequence are in fact frequent 1-sequences, the algorithm generates a list of candidate 2-sequences by concatenating all frequent 1-sequences to one another. It then counts all candidates by iterating the entire dataset, and maintains a list of frequent 2-sequences. The algorithm then generates a list of candidate 3-sequences based on combinations of frequent 2-sequences, and the process repeats until no more frequent patterns are discovered. The main advantage of frequency-based techniques is the fact that many sequences that appear in the dataset are not counted at all. This is due to the *closure* property of the frequency measure, i.e. the fact that each subsequence of a frequent pattern is also frequent. Consequently, frequency-based techniques are appropriate for analyzing and learning from large datasets, such as transactional databases of retail organizations [22].

However, frequent sequential patterns are not necessarily meaningful. For instance, if an event a is frequent, and an event b is frequent, then the events ab, ba are often frequent as well, simply due to random chance which places them next to each other (random co-occurrence). To battle such spurious frequent patterns generated by the first-tier support-based algorithm, many investigations propose a number of second-tier methods, to further filter the results generated by the first tier.

A common second-tier technique—*confidence*—measures the likelihood of a sequence's suffix given its prefix, i.e., the predictive power of the prefix with respect to the suffix. For a sequence p composed of a prefix p_r, the confidence of p is given by $conf(p) = \frac{freq(p)}{freq(p_r)}$. The Support/Confidence framework ([2]) detects all sequences whose frequency exceeds a user defined *min_support* and confidence exceeds a user defined *min_confidence*.

The Support/Confidence approach suffers from several specific limitations. First,

	$COFFEE$	$\neg COFFEE$	
TEA	20	5	25
$\neg TEA$	70	5	75
	90	10	100

Table 1: Correlation between purchases of tea and coffee.

significant patterns are not necessarily very frequent. Reducing the minimal support threshold might help, but this also introduces many uninteresting patterns and affects the performance of the learning process [1]. A second problem ([1, 12, 23]) is the flooding of results, i.e. the large number of patterns returned by the algorithm. The problem is emphasized as the data set becomes more dense [1]. One way to address the flooding challenge is to rely on external knowledge to filter irrelevant patterns. This knowledge can either be explicitly provided by a domain expert ([12]) or inferred from the domain, e.g. when sequences are labeled ([23]). These solutions, however, require knowledge outside of the original unlabeled data.

The third problem is the spuriousness of the results [1, 5, 18]. Silverstein et. al [18] demonstrate the problem with the following example. Consider a database containing information about goods that have been bought together by customers. In particular, consider a candidate association rule that associates purchases of tea and coffee. The relevant data is shown in Table 1. For instance, the table shows (row 1, marked TEA) that out of 25 customers that bought tea, 20 also bought coffee, while 5 did not. Also (column 2, marked ¬COFFEE), out of 10 customers that *did not* buy coffee, 5 bought tea (row 1), and five did not.

The strength of the rule TEA $\rightarrow$ COFFEE can be calculated by the Support/Confidence approach using the contingency table (Table 1). The support for this rule is 20%, which is quite high, and its confidence is 80%, certainly a high value, as 20 out of 25 tea buyers have also purchased coffee. Consequently, the rule TEA $\rightarrow$ COFFEE is expected to be part of the results set. Note, however, that the apriori probability of a customer to purchase coffee is 90% (margin of first column). In other words, a customer that buys tea is *less likely* to buy coffee than a customer we don't know anything about. Moreover, a customer who doesn't buy tea is much more likely to purchase coffee. This means there is in fact a *negative* correlation between tea and coffee, although the rule TEA $\rightarrow$ COFFEE would have been discovered by the Support/Confidence framework. This results from the fact that the Support/Confidence framework does not consider the null hypothesis, i.e. the chance to encounter the pattern under the assumption of independence.

2.2 Statistical and Probability Approaches

In direct response to the specific deficiencies of support/confidence techniques, alternative statistical and probability-based analysis methods have been proposed, typically to replace the confidence measure. [21] surveys a large number of such measures and demonstrates that they can provide conflicting information.

Interest is a measure that attempts to contrast predictive power with the probabilistic independence assumption [5, 18, 21]. Let AB be an sequence, where both A and B reflect patterns containing one or more events. Then the interest measure of AB is $\frac{sup(AB)}{sup(A)sup(B)}$. Interest values above 1 indicate positive dependence, while those below 1 indicate negative dependence. Thus in contrast to confidence, interest considers the assumption of independence of A and B, and might therefore provide better results as a second-tier technique (on top of support).

However, previous work has recognized limitations specific to interest [18]. In particular, since interest does not consider the frequency of A and B, its absolute value may be misleading. For instance, long patterns, which are less expected to appear together under the assumption of independence, almost always receive very high values of interest despite the fact that their frequency is typically very low and they might therefore be insignificant. Indeed, [18] suggests that interest should only be used to compare different associations between set of events that are already considered correlated, and not for comparing different patterns. Another possible problem is that interest is completely symmetric, which makes its use in sequence learning applications (where $A \rightarrow B$ and $B \rightarrow A$ are different) problematic.

To address these difficulties, *conviction* has been proposed in [5]. It is defined as $\frac{sup(A)sup(\neg B)}{sup(A\neg B)}$, with the underlying intuition that $A \rightarrow B$ is in fact $\neg(A \wedge \neg B)$, and thus we can measure how far $A \wedge \neg B$ deviates from independence, and then invert the ratio to handle the negation. Unlike interest, conviction is asymmetric and is therefore more appropriate for measuring sequences. However, conviction still suffers from similar flaws as interest in other ways. For instance, like confidence, conviction does not take frequency into account. Thus, two events A and B that have occurred only once and happened to appear together are given the maximal conviction rate of ∞. Also, like interest, conviction favors longer sequences, which are less expected to appear under the assumption of independence (this is indeed mentioned in [5], which claims many rules were very long and too complicated to be interesting).

A different approach is presented in [1]. The authors present a measure called *collective strength*, which is defined as follows. For each item-set I in the data set the violation rate of the item, denoted $v(I)$, is the fraction of transactions that contain some, but not all, of the attributes of I. The collective strength of an item I is defined as $C(I) = \frac{(1-v(I))E[v(I)]}{v(I)(1-E[v(I)])}$, where E reflects expectation under the assumption of independence. Note that the collective strength increases as the number of "violating" transactions decreases, and as the number of "supportive" transactions exceeds expectations. The value of C(I) under the assumption of independence is 1, and it exceeds this value in case of an interesting pattern. However, this measure can also be quite misleading, as mentioned in [7], in that strong statistical correlations between relatively infrequent attributes yield collective strength values that only marginally exceed 1, the value of $C(I)$ under the assumption of independence. For instance, assuming attributes A, B, C and D each appear in 5% of the itemsets, they are all expected to appear together in less than 0.0006% of the itemsets (0.05^4) and are expected not to appear at all in $\approx 0.81\%$ of the itemsets. If all attributes appear together in 1% of the itemsets, which is ≈ 1666 times more than what is expected under the assumption of independence, and all do not appear at all in 0.82% of the itemsets (which matches expectations), the

collective strength of $ABCD$ is ≈ 1.11, only slightly above the independence value.

Moreover, the definition of a strongly collective item-set requires not only a high collective strength rank, but also demands all sub-itemsets to be strongly collective. This definition makes sure the suggested measure maintains the closure property, and makes it appropriate for large-scale data mining purposes. However, this also means that strongly-collective itemsets contain items which are all correlated to one another. This property is not guaranteed in all data mining domains. For instance, one may want to discover whether a certain combination of age, gender and origin increases the chance of getting a certain disease. A rule such as $Age > 80, Male, European \rightarrow AlzheimerDisease$ may, of course, be interesting, although there is no correlation between the attributes reflecting age, gender and origin, and therefore it will not be considered strongly collective.

Silverstein et. al [18] offer an extensive discussion of the relative weaknesses and strengths of different statistical approaches, and suggests learning association rules that are judged significant using a chi-squared statistical test. For each k-item-set (a set of k attributes within a customer basket), the authors run a chi-squared statistical test, using a k-dimensional contingency table, where each cell reflects a combination of the k attributes involved. For instance, for the pattern ABC the authors suggest counting ABC, $AB\neg C$, $A\neg BC$, $A\neg B\neg C$, $\neg ABC$, $\neg AB\neg C$, $\neg A\neg BC$ and $\neg A\neg B\neg C$. The authors claim that chi-square test is upward closed. In other words, for each 3 attributes A, B and C $chi - square(ABC) >= chi - square(AB)$. Based on this property the suggested algorithm stops mining patterns that already contain significant subsequences, assuming only *minimally dependent* association rules are interesting. However, later work [7] has shown that although raw chi-square values are indeed increasing when expanding a rule by additional attributes, the statistical significance (p-value) does not necessarily increase. This invalidates the concept of minimally dependent association rules.

A similar technique uses a G-test [19] for detecting statistically significant patterns [10, 15]. To calculate the rank of a given pattern p, a 2×2 contingency table is built for its prefix p_r and suffix α_k (Table 2). In the top row, n_1 is the number of times that we saw the pattern p (p_r followed by α_k), and is simply $freq(p)$. n_2 is the number of times we saw a different suffix to the same prefix, i.e., $\sum_{\alpha_i \neq \alpha_k} freq(p_r \alpha_i)$. In the second row, n_3 is the number of patterns in which α_k followed a different prefix than p_r ($\sum_{p_m \neq p_r} freq(p_m \alpha_k)$). n_4 is the number of patterns in which a different prefix was followed by a different suffix ($\sum_{p_m \neq p_r} \sum_{\alpha_i \neq \alpha_k} freq(p_m \alpha_i)$). The table margins are the sums of their respective rows or columns. A G-test is then run on the contingency table to calculate the dependence of α_k on p_r as follows: $G = 2 \sum n_i \times \log \frac{n_i}{E_i}$, where E is the expected frequency under the assumption of independence.

The advantage of dependency detection methods (hereinafter, marked *DD*) such as chi-square or G-test over other statistical measures such as interest, conviction and collective strength is that they consider both frequency and departure from independence. Another advantage of these tests is that while they do not maintain the closure property, it is still possible to compute an upper bound for their ranks. Such an upper bound evaluation was used in [15] as a pruning technique for searching the best correlations between multiple streams of data. DD methods have been utilized in several data analysis applications, including analysis of execution traces [10], time-series analysis [6],

	α_k	$\neg\alpha_k$	
p_r	n_1	n_2	$freq(p_r)$
$\neg p_r$	n_3	n_4	$\sum_{p_m \neq p_r} freq(p_m)$
	$freq(\alpha_k)$	$\sum_{\alpha_i \neq \alpha_k} freq(\alpha_i)$	

Table 2: A statistical contingency table for sequential pattern p, composed of a prefix $p_r = \alpha_1, \alpha_2, \ldots, \alpha_{k-1}$ and suffix α_k.

and RoboCup soccer coaching [11]. However, DD methods have not been shown to scale to large databases, and (as we show below) also suffer from a length-based bias.

2.3 Summary

The problem of the poor quality of the results generated by existing methods is well known in the literature, as the discussion above demonstrates. However, investigations have often tended to point out specific limitations of previous work. Thus work on interest is largely motivated by limitations of confidence, and work on conviction is motivated by limitations of interest, etc.

This paper focuses instead on addressing general limitations: Those that are common to many or all approaches. The next section begins with an empirical investigation of two representative approaches, and draws conclusions as to such general limitations.

3 Sequential pattern learning techniques: Initial experiments

Our research begins with a comparison of several sequence mining techniques, focusing on the quality of their results. Our main interest is to contrast the results of support-based and statistical techniques, and their combinations, to find out whether one of these approaches outperforms the other significantly and under which conditions. Section 3.1 describes the experiments we performed on synthetic data. Our experiments using real-world data are described in section 3.2.

3.1 Synthetic Data Experiments

We conducted extensive experiments using synthetic data, comparing Support, Confidence, Support/Confidence and dependency-detection using a G-test (DD). In each run, these techniques were to discover five different re-occurring *true segments*, uniformly distributed within a file of 5000 streams. We refer to the percentage of the streams that contain true segments as *pattern rate*; thus low pattern rates indicate high levels of noise. The example streams might include additional random events before, after, or within a segment. We controlled *intra-pattern noise rate*: The probability of having noise inserted within a pattern. However, in the following tests no noise was inserted (we evaluate the effects of intra-pattern noise in Section 5).

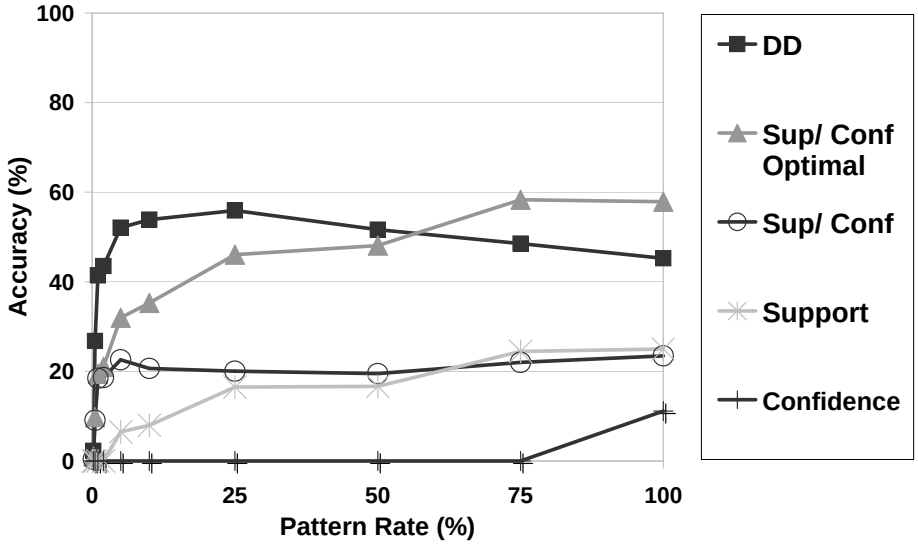


Figure 1: Accuracy of unsupervised sequence learning methods.

In each experiment, each technique reported its best 10 segment candidates, and those were compared to the five true segments. The results were measured as the percentage of true segments that were correctly detected (the recall of the technique, hereinafter denoted *accuracy*). The Support/Confidence technique requires setting manual thresholds. To allow this method to compete, we set its thresholds such that no true pattern would be pruned prematurely. We refer to this technique as "Support/Confidence Optimal". We have also tested a more realistic version of the algorithm, using fixed minimal confidence of 20% ("Support/Confidence"). While the Support/Confidence method is meant to return all segments satisfying the thresholds, with no ordering, we approximated ranking of resulting segments by their support (after thresholding).

We varied several key parameters in order to verify the consistency of the results. For three different values of alphabet size, denoted T (5, 10 and 26) and three ranges of true-pattern sizes (2–3, 3–5 and 4–7) we have generated data sets of sequences with incrementing values of pattern rate. For each pattern rate we have conducted 50 different tests. Overall, we ran a total of 4500 tests, each using different 5000 sequences and different sets of 5 true patterns.

The results are depicted in Figure 1. The X-axis measures the pattern rate from 0.2% to 100%. The Y-axis measures the average accuracy of the different techniques over the various combinations of T and pattern size. Each point in the figure reflects the average of *450* different tests. The "Optimal Support/Confidence" technique is denoted "Sup/Conf Optimal", where the standard method, using a fixed minimal confidence value, is denoted "Sup/Conf". The dependency-detection is denoted "DD".

The figure shows that dependency-detection (*DD*) outperforms all other methods for low and medium values of pattern rate. However, the results cross over and

Support/Confidence optimal outperforms DD at high pattern rates. This unexpected degradation in DD results at increasing pattern rates is addressed in Section 4.1. The manually-set Support/Confidence, as well as the simple support technique, provide relatively poor results. Finally, confidence essentially fails for most pattern rate values.

Figure 2 shows the results for the same experiment, focusing on pattern rates up to 5%. As can be clearly seen, DD quickly achieves relatively high accuracy, at least twice as accurate as the next best technique, Support/Confidence optimal. A paired one-tailed t-test comparing DD and Support/Confidence optimal for pattern rates of up to 5% shows that the difference is significant at the 0.01 significance level ($p < 1 \times 10^{-10}$).

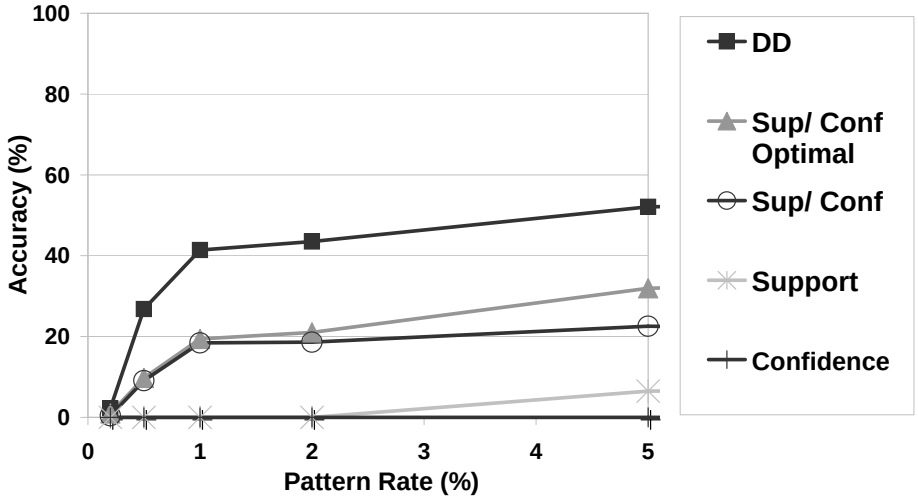


Figure 2: Accuracy at low pattern rates (high noise).

We evaluated the effect of alphabet size T on the accuracy of the different algorithms. Figures 3 and 4 show the accuracy we measured for the two different alphabet sizes 5 and 26 respectively, corresponding to the tests depicted in Figure 2. The alphabet size is concatenated to the name of each technique, e.g., "DD-26" corresponds to the results of the DD technique for an alphabet of size 26. The Figures clearly show that all methods have achieved better results when a larger alphabet was used. The reason for this is that as the size of the alphabet grows, the chance of the noise to form significant patterns in terms of the tested techniques decreases.

More importantly, the advantage of DD over the other techniques in high noise settings becomes even more significant for the smaller alphabet tests. For $T = 5$ DD's results are 25 times more accurate than the second best technique for pattern rate of 0.5%, and more than 6 times more accurate for pattern rates between 1% and 5%. Similar results were achieved in experiments where the size of true patterns was varied.

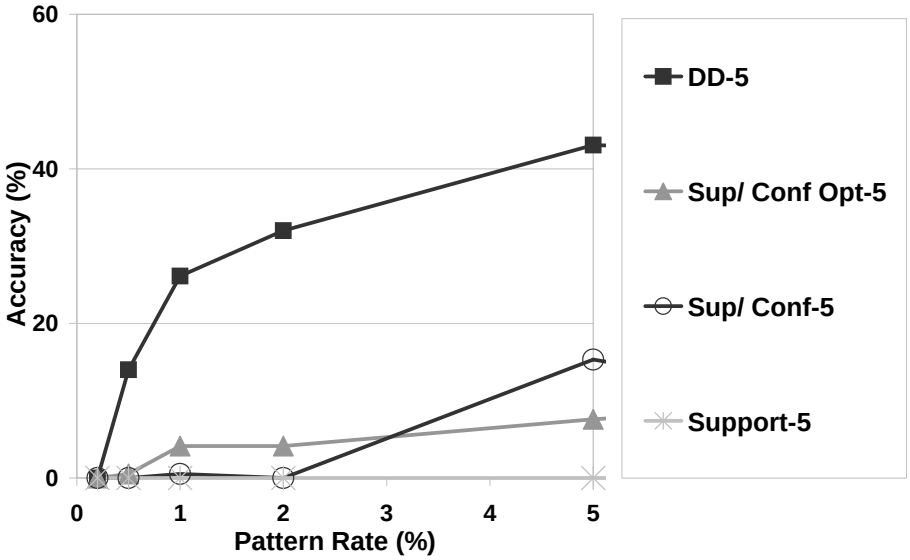


Figure 3: Accuracy for alphabet size T=5.

3.2 Real World Experiments

We conducted real-world experiments on UNIX command line sequences. We utilized 9 data sets of UNIX command line histories, collected for 8 different users at Purdue university over the course of 2 years [8]. In this case, we do not know in advance what true patterns were included in the data, thus quantitative evaluation of accuracy is not possible. However, we hoped to qualitatively contrast the pattern candidates generated by the different methods. The data has been stripped out of file names, user names etc., so only command names and flags remained. The arguments of each command were replaced by a token that reflects their number. For instance "ls -la /private/docs" was replaced by "ls -la <1>" and "cat foo.txt bar.txt zorch.txt > somewhere" was replaced by "cat <3> > <1>". We applied a minimal confidence threshold of 0.25 for the Support/Confidence technique, and used G-test for DD.

The results imply once again that DD is superior to the other tested methods, but its advantage in this case is less obvious. For simplicity we focus on the results of one user. We begin with the first 10 patterns discovered by Support/Confidence, depicted in Table 3. For each sequence the table shows its frequency and confidence, as well as its position within the DD results list.

The results depicted in Table 3 consist of very short sequences, reflecting the most common Unix commands, such as ls, cd, vi and $more$. The only interesting result is "f <1>", because we are not familiar with the Unix command "f" ("f" is probably an alias that our specific user has configured for one of the common shell commands).

This tells us something about the nature of frequency-based results, which can be useful for learning the most basic sequential behavior of a user, but might fail in

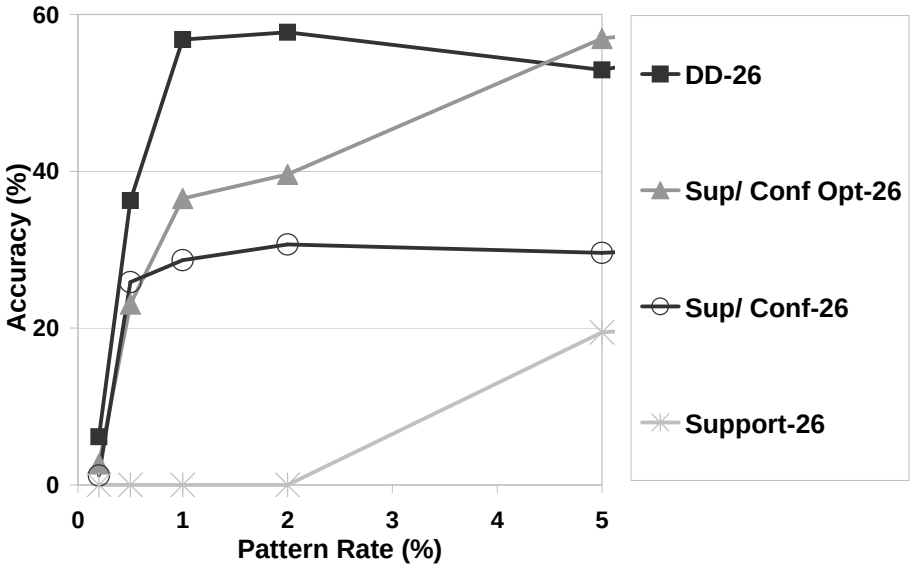


Figure 4: Accuracy for alphabet size T=26.

detecting more complex / interesting phenomena. For someone who is not familiar with Unix (like we are not with the command "f") the results can be very useful, because they point at the most common features in a new domain. However, for someone who is looking for more novel information, the results might be disappointing. A Similar point was made in [5] with respect to highly ranked conviction-based rules that were extracted from Census data and included examples such as "five year olds don't work", "unemployed residents don't gain income from work" and "men don't give birth". Given an adult's perspective, such rules are not interesting. But to someone (say, a theoretical alien), these results may in fact be interesting.

Interestingly, Table 3 also shows that the most frequent sequences in the dataset have also received high DD ranks. In fact, 7 out of the 8 most frequent sequences were included in the top 8 results of DD. This seems surprising, because rules such as "cd <1>" and "vi <1>" do not seem to be statistically significant, as the existence of a single argument does not seem highly correlated with the commands cd or vi.

The reason for this behavior is that DD, like many other methods we tested, is biased towards frequent sequences, which yield high-confidence rules. Given two statistically significant patterns with the same underlying relations between n_1, n_2, n_3 and n_4 in the contingency tables, the more frequent pattern will also receive a higher score by statistical tests such as G or Chi-Square. In order to understand this better, let us examine the contingency table of the sequence "cd <1>" (Table 4). Out of the 987 appearances of "cd" it was followed by a single argument 832 times (84% of the cases). Among all other 14145 2-sequences, only 3450 (24%) were followed by the single argument indicator "<1>". The fact that "cd" is a relatively frequent command also helps

Sequence	Frequency	Confidence	SC position	DD position
<1>; ls	1088	0.25	1	7
cd <1>	832	0.84	2	3
vi <1>	774	0.99	3	1
more <1>	669	0.83	4	6
cd <1>; ls	607	0.73	5	2
<1>; vi <1>	464	0.99	6	4
ls; cd	430	0.25	7	13
f <1>	404	0.98	8	8
ls; cd <1>	364	0.85	9	18
<1>; ls; cd	309	0.28	10	20

Table 3: User sequences with highest support

	<1>	¬ <1>	
cd	832	155	987
¬ cd	3450	10695	14145
	4282	10850	15132

Table 4: Contingency table for "cd <1>"

rejecting the null hypothesis and establishing the statistical correlation between "cd" and "<1>". For instance, if "cd" had appeared only 10 times, and was followed by a single argument in 9 of the cases (maintaining a similar ratio to our database, and leaving all other sequences unchanged) the G rank of "cd <1>" would have been ≈ 20, comparing to ≈ 1450, the original rank of "cd <1>".

By now we have covered the 10 most frequent sequences in our dataset. Looking further down the result list reveals the advantage of DD over the Support/Confidence approach. Table 5 depicts the 10 next candidates of Support/Confidence, where table 6 shows the 10 next candidates of DD, starting from candidate 5 (which was skipped before due to its relatively low frequency) and skipping other frequent candidates we have already covered. For each sequence we indicate its position within both Support/Confidence and DD result lists, or '-' when it is not included in the top 20 sequences of the relevant technique.

The sequences in Table 5 are not of much interest. Most of the sequences represent different combinations of frequent commands such as ls, cd and $more$. Note that most of these irrelevant sequences are not detected by DD, despite their high frequency and confidence values. The only news we learn from Table 5 are two new frequent one-argument shell commands — rm and gcc, one of which is also discovered by DD.

Table 6 on the other hand, contains a lot of interesting information. The first command—"q -v"—received a very high rank by DD, although it is relatively infrequent. Once again, we are not familiar with the Unix command "q" and thanks to DD we learn not only about its existence, but also about its correlation with the flag "-v" (we

Sequence	Frequency	Confidence	SC position	DD position
rm <1>	297	0.92	11	19
ls; cd <1>; ls	287	0.79	12	9
<1>; ls; cd <1>	261	0.84	13	-
ls; more <1>	235	1.00	14	-
cd <1>; ls; cd	219	0.36	15	-
gcc <1>	212	0.99	16	-
<1>; ls; cd <1>; ls	201	0.77	17	17
<1>; more <1>	196	0.99	18	-
cd <1>; ls; cd <1>	189	0.86	19	-
<1>; cd <1>	189	0.84	20	-

Table 5: Support/Confidence: 10 next user sequences.

Sequence	Frequency	Confidence	SC position	DD position
q -v	99	0.94	-	5
ls; cd <1>; ls	287	0.79	12	9
mv <2>	125	0.93	-	10
g++ -g	94	0.47	-	11
| more	136	0.98	-	12
gcc <1>; a.out	144	0.68	-	14
ls -al	160	0.09	-	15
<1>; gcc <1>; a.out	125	0.68	-	16
<1>; ls; cd <1>; ls	201	0.77	17	17
ls; cd <1>	364	0.85	9	18

Table 6: DD: 10 next user sequences.

assume this is an alias for the "quota" command). In the 11th position we learn about another relatively infrequent command, "g++", which is used in sequence with the flag "-g". The 12th sequence discovers the non-trivial correlation between the pipe symbol and the shell command "more". The 14th sequence discovers an even more interesting sequential behavior — the correlation between compilation ("gcc <1>") and execution ("a.out"). The 15th sequence reveals the correlation between the command "ls" and the flags "-al". Interestingly, the relatively infrequent command "ls -al" also has a low confidence, and would therefore be completely excluded from the Support/Confidence results list. The reason for this is that the flags "-al" followed the command "ls" in only 160 out of 1710 executions ($\approx 9\%$). DD has discovered this sequence because out of the other 13422 2-sequences, which do not start with the command "ls", the flags "-al" have appeared only once. This helped establishing the correlation between "ls" and "-al" despite the low confidence. Finally, DD has also discovered the correlation between the command "mv" and the two arguments that often followed it (the 10th sequence in DD list, "mv <2>"), since most other shell commands in the dataset were followed by either zero or one arguments.

In conclusion, after skipping the most frequent sequences in the dataset, which receive the highest ranks from both Support/Confidence and DD, DD uncovers interesting sequential behavior while Support/Confidence results contain mostly irrelevant combinations of frequent events. A somewhat similar notion of discovering interesting results "down the list" was suggested in [5] for conviction-based rules on Census data.

4 Statistical Biases

Following the empiric comparison, we analyze the failures and successes of the different techniques. While DD was significantly better than Support/Confidence, it did not necessarily fair particularly well on an absolute scale, especially taking into account its seemingly-paradoxical drop in performance with higher pattern rates. On the other hand, support-based methods, even involving confidence, did not appropriately handle low pattern rates (high noise settings). Moreover, although DD was able to extract some useful sequential information from our real-world datasets, all methods have essentially failed to detect more complex sequential behavior, such as significant correlations between two or three different shell commands.

We have found that all methods suffer from common limitations: (i) a bias in the ranking, based on length (Section 4.1); (ii) inability to generalize from similar discovered patterns (Section 4.2).

4.1 Removing the Length Bias

The first common limitation of the approaches described above is their bias with respect to the length of the segments. Figure 5 shows the average length of the segments returned by the learning algorithms, in a subset of the tests shown in Figure 1, for two different values of pattern rate (0.5% and 75%), where the length of the true patterns was set to 3–5 (average ≈ 4) and alphabet size was fixed at 10. The figure shows that the support algorithm prefers short segments. The optimal Support/Confidence

algorithm behaves similarly, though it improves when pattern rate increases (75%). DD is slightly better, but also prefers shorter sequences at low noise (high pattern rate) settings. In contrast to all of these, Confidence prefers longer sequences.

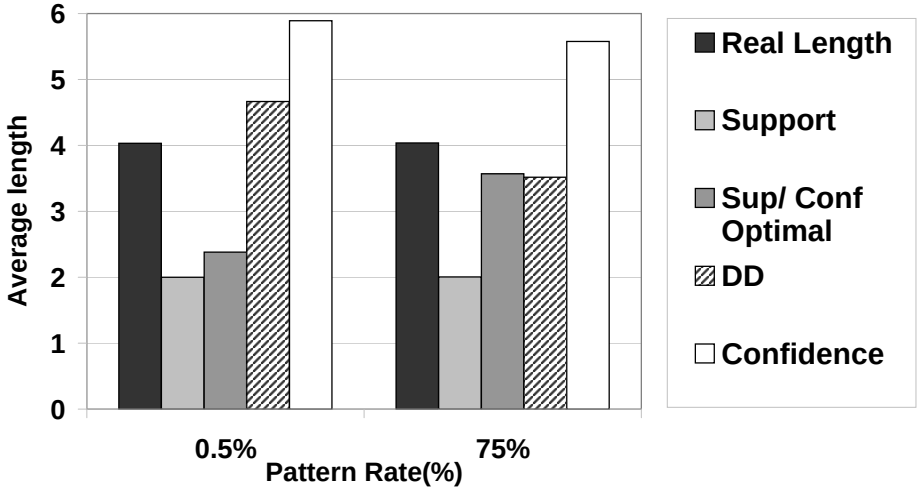


Figure 5: Average segment length for two pattern rate values.

Different methods have different reasons for these biases. Support-based methods have a bias towards shorter patterns, because there are more of them: Given a target pattern *ABCD*, the pattern *AB* will have all the support of *ABCD* with additional support from (random) appearances of *ABE*, *ABC*, *ABG*, etc. Confidence has a bias towards longer sequences, because their suffix can be easily predicted based on their prefix simply because both are very rare. Finally, DD methods prefer shorter segments at higher pattern rate settings. We found that this is due to DD favoring subsequences of true patterns to the patterns themselves. When pattern rate is high, significant patterns also have significant sub-patterns. Moreover, the sub-patterns may have higher significance score because they are based on counts of shorter sequences—which are more frequent as we have seen. This explains the degradation in DD accuracy at higher pattern rates.

In order to overcome the length bias obstacle, we normalize candidate pattern ranks based on their length. The key to this method is to normalize all ranking based on units of standard deviation, which can be computed for all lengths (i.e., their standard *z* score). Given the rank distribution for all candidates of length k, let $\bar{R}^k$ be the average rank, and $\hat{S}^k$ be the standard deviation of ranks. Then given a sequence of length k, with rank r, the normalized rank will be $\frac{r-\bar{R}^k}{\hat{S}^k}$. This translates the rank r into units of standard deviation, where positive values are above average. Using the normalized rank, one can compare pattern candidates of different lengths, since all normalized ranks are in units of standard deviation. This method was used in [6] for unsupervised segmentation of observation streams based on statistical dependence tests. To our best knowledge, this is a first application in sequence learning, and in the context of methods

other than DD.

4.2 Generalizing from Similar Patterns

A second limitation we have found in all evaluated methods is inability to generalize patterns, in the sense that sub-segments of frequent or statistically significant patterns are often themselves frequent (or significant). Thus both segment and its subsegment receive high normalized ranks, yet are treated as completely different patterns by the learning methods. For instance, if a pattern $ABCD$ is ranked high (after normalization), the algorithm is likely to also rank high the *shadow sub-patterns* ABC, BC, etc. Normalizing for length helps in establishing longer patterns as preferable to their shadows, but the shadows might still rank sufficiently high to take the place of other true patterns in the final pattern list.

We focus on a clustering approach, in which we group together pattern variations based on their edit distance. Clustering techniques are widely used in the data mining community. Most related to our work are [4] and [13], which use clustering techniques to learn the sequential behavior of users. A common theme is that clustering is done based on similarity between sequential pattern instances in the training data. Bauer [4] uses the resulting clusters as classes for a supervised learning algorithm. Lane and Brodley [13] use the clusters to detect anomalous user behavior. Neither investigates possible statistical biases as we do.

We cluster candidates that are within a user-specified threshold of *edit distance* from each other. The procedure goes through the list of candidates top-down. The first candidate is selected as the representative of the first cluster. Each of the following candidates is compared against the representatives of each of the existing groups. If the candidate is within a user-provided edit-distance from a representative of a cluster, it is inserted into the representative's group. Otherwise, a new group is created, and the candidate is chosen as its representative. The result set is composed of all group representatives.

Generally, the edit-distance between two sequences is the minimal number of editing operations (insertion, deletion or replacement of a single event) that should be applied on one sequence in order to turn it into the other. For example, the editing distance between ABC and ACC is 1, as is the editing distance between AC and ABC. A well known method for calculating the edit distance between sequences is *global alignment* [17].

However, our task requires some modifications to the general method. For example, the sequence pairs $\{ABCDE, BCDEF\}$ and $\{ABCD, AEFD\}$ have an edit-distance of 2, though the former pair has a large overlapping subsequence ($BCDE$), and the latter pair has much smaller (fragmented) overlap $A??D$.

We use a combination of a modified (weighted) distance calculation, and heuristics which come to bear after the distance is computed. Our alignment method classifies each event (belonging to one sequence and/or the other) as one of three types: appearing before an overlap between the patterns, appearing within the overlap, or appearing after the overlap. It then assigns a *weighted* edit-distance for the selected alignment, where the edit operations have weights that differ by the class of the events they operate on. Edit operations within the overlap are given a high weight (called *mismatch*

weight). Edit operations on events appearing before or after the overlap are given a low weight (*edge weight*). In our experiments we have used an infinite mismatch weight, meaning we did not allow any mismatch within the overlapping segment. However, both weight values are clearly domain-dependent. We leave further investigation of this issue for future work.

In order to avoid false alignments where the overlapping segment is not a significant part of the overall unification, we set a minimal threshold upon the length of the overlapping segment. This threshold is set both as an absolute value and as a portion of the overall unification's length.

5 Experiments

To evaluate the techniques we presented, we conducted extensive experiments on synthetic (Section 5.1) and real data (5.2). We show significant improvements to all methods.

5.1 Synthetic Data Experiments

We repeated our experiments from Section 3, this time with the modified techniques. Figures 6 and 7 show the accuracy achieved at different pattern rates, paralleling Figures 1 and 2, respectively. The figures contrast standard, normalized (marked *N*) and normalized-clustered (*NC*) versions of DD, Support, and Optimal Support/Confidence (called simply Support/Confidence henceforth).

Figure 6 shows the results for all pattern rates, separately for support (Figure 6-a), Support/Confidence (6-b), and DD (6-c). Figure 7 focuses on low pattern rates (high noise) for the same techniques. Every point in the figures is the average of *450* different trials.

The results show that length normalization improves *all* tested learning methods, sometimes dramatically. For instance, the original support technique completely fails to detect true segments for pattern rate of 1%. However, its normalized version achieves accuracy of 39% at this rate. Clustering the normalized results improved the results further, by notable margins.

The improvements derived from normalizing and clustering the results both proved to be statistically significant for all learning techniques. For instance, a paired one-tailed t-test shows that the normalized version of DD is significantly better than the standard version ($p < 1 \times 10^{-10}$) and that the clustered-normalized version of DD significantly outperforms the normalized version ($p < 1 \times 10^{-10}$).

The improvements are such that after normalization and clustering, the simple support technique outperforms the standard DD method for all pattern rate values, including 0.2%-0.5%. This is where standard DD performs significantly better than the other standard techniques. Indeed, after length-based standardization and clustering, DD may no longer be superior over the Support/Confidence approach for any range of pattern rate.

Figure 8 shows the results from one specific setting, where both normalizing and normalizing-clustering proved particularly effective. Each point in the figure represents

the average of 50 different tests, with an alphabet size 26, and true patterns composed of 3–5 events. In the figure, the normalized version of the support technique has achieved accuracy of 78% for a pattern rate of 1%, comparing to 0% accuracy of the standard version. The Normalized-Clustered versions of all algorithms have achieved more than 95% accuracy for a pattern rate as low as 1%, where the accuracy of the standard techniques was 0% for support, 66% for Support/Confidence and 82% for DD.

These non-trivial results (in particular for support/confidence techniques) stem from the fact that some of the major deficiencies we described in section 2.1 with respect to the Support/Confidence approach are inhibited by our normalization and clustering techniques. In particular, short frequent sequences, which previously flooded the results returned by the technique, are suppressed by our length-based normalization. Moreover, the techniques also improve the ability of Support/Confidence methods to handle significant, yet infrequent sequences, since long significant sequences are often relatively frequent, when comparing them to other sequences of the same length. They are not detected by the standard Support/Confidence framework, but after normalization and clustering they are often assigned high ranks. However short significant yet infrequent sequences (like the "quota -v" example from section 3.2), are not discovered even by the clustered normalized frequency-based techniques.

We now turn to examine the effects of intra-pattern noise on the quality of the result segments. Figure 9 shows the accuracy resulting from using the standard, Normalized and Normalized-Clustered methods separately for support (Figure 9-a), Support/Confidence (9-b), and DD (9-c). Pattern-rate was fixed at 5%, while we varied the intra-pattern noise rate from 0% to 50% (on the X axis). As expected based on the results presented above, the figures show a distinct advantage to using clustering with all normalized methods. In addition, the figures show that this advantage is maintained even when the amount of intra-pattern noise is increased, although it becomes less significant. The results in Figure 9 were measured for an alphabet of 26 events and true patterns of size 3–5. We have measured similar results for all settings of T and pattern size.

The ability to measure the improvements stemming from the use of normalization and clustering technique is of course a function of the ability to control the data. However, we wanted to go beyond purely synthetic data.

We thus evaluated our techniques with an additional dataset. We used the text of George Orwell's *1984* to test our modified techniques on data that was both realistic, yet still allowed for controlled experiments. In one experiment, we changed the original text by introducing noise within the words and between them. For instance, the first sentence in the book - "It was a bright cold day in April" was replaced by "**It**oH7l4H XywOct8M (...9 more noisy words) 6jO**was**2x imfG8e1x (...2 more noisy words) nB**a**or1oL iWtHhTEq **bright**cT x**cold**Vuv vf**day**1Ap BsQG9pyK 8Nxf**in**XR 8TGmxcXO E1IenU2Q **Apri**u**l**xL".

We inserted only fixed 8-character sequences, such that each actual word that is shorter than 8 characters was padded with noise, and words longer than 8 characters were cut. We set pattern rate to 40% by inserting 6 noisy streams, for each 4 containing actual words. Intra pattern noise was set at 10%. We then counted how many of the top 100 candidates returned by each technique are actual words appearing in the book. We hoped to find as many actual words in the results set as possible. The results, reflecting

the average accuracy over the first 8 chapters of the book, are shown in Figure 10. Similar improvement results were achieved for other settings of pattern rate and intra pattern noise.

The results show that for each of the presented techniques the Normalized-Clustered versions have significantly outperformed the standard versions, increasing accuracy by up to 41% for the support algorithm. The normalized versions have typically outperformed the standard versions, except for the case of DD, where the normalized results contained various sequences that reflected the same words (see Section 4.2), and were then significantly improved by our clustering approach. Note also that among the standard techniques, DD has once again outperformed the other methods.

Tables 7 and 8 show the top 25 sequences of Support/Confidence (SC) and normalized-clustered Support/Confidence (SC-NC) respectively, as measured on the first chapter of the book. For each sequence each table contains its frequency and confidence, as well as its position within the results list. Table 8 also contains the original position of each sequence within the results list of the standard Support/Confidence technique. The results demonstrate the difference between the standard results and the Normalized-Clustered ones. While the results of standard Support/Confidence are mostly composed of short sequences, the normalized-clustered results are able to detect more complex sequential patterns, such as the name of the book's main character Winston, which appears second in the normalized-clustered results list, and only 71st in the original list.

5.2 Experiments with Real-World Data

We now turn back to our real-world UNIX command-line sequences, as described in section 3.2. Tables 9 and 10 depict the results of the top 10 sequences returned by the Normalized-Clustered techniques of Support/Confidence (SC) and DD respectively. The results were received on the same user dataset which served us for the demonstration of Tables 3, 5 and 6. For each user sequence the tables contain the following fields.

1. Support.
2. Confidence.
3. Position within the Normalized-Clustered results list of the given technique.
4. Position within the original results list of the given technique.
5. Position within the Normalized-Clustered results list of the alternative technique. Due to clustering the result within the alternative list might not be identical to the given sequence.
6. Position within the original results list of the alternative technique.

The results of Tables 9 and 10 suggest that the clustered-normalized versions of both DD and Support/Confidence discovered similar sequential patterns. This was the case for most of the result sequences. More importantly, both techniques detected

Sequence	Frequency	Confidence	SC position
he	799	0.23	1
the	466	0.58	2
ing	220	0.36	3
was	183	0.63	4
and	162	0.46	5
ere	106	0.22	6
hat	91	0.30	7
ent	89	0.25	8
ver	86	0.43	9
had	80	0.26	10
his	79	0.28	11
ith	73	0.25	12
that	67	0.78	13
eve	67	0.60	14
wit	67	0.55	15
all	66	0.37	16
ter	65	0.25	17
with	57	0.85	18
ome	57	0.33	19
een	53	0.31	20
ove	52	0.60	21
nst	52	0.47	22
The	51	0.61	23
here	50	0.37	24
ess	50	0.21	25

Table 7: 1984: Support/Confidence: Top 25 sequences.

Sequence	Frequency	Confidence	SC-NC position	SC position
the	466	0.58	1	2
Winston	31	1.00	2	71
ing	220	0.36	3	3
that	67	0.78	4	13
was	183	0.63	5	4
with	57	0.85	6	18
and	162	0.46	7	5
screen	23	0.92	8	115
here	50	0.37	9	24
which	29	0.97	10	86
moment	21	1.00	11	127
hough	27	1.00	12	91
Goldstei	15	1.00	13	197
ight	43	0.98	14	40
telescre	14	1.00	15	235
ould	38	0.93	16	48
Brother	14	1.00	17	235
ever	35	0.52	18	55
tion	34	0.83	19	58
Writing	13	1.00	20	270
Ministry	11	0.92	21	362
possible	11	1.00	22	362
this	17	0.27	23	161
O’brien	12	1.00	24	319
because	12	1.00	25	319

Table 8: 1984: Normalized-Clustered Support/Confidence: Top 25 sequences.

Sequence	Freq	Conf	NC-SC pos	SC pos	NC-DD pos	DD pos
cd <1>; ls; cd <1>; ls	140	0.74	1	29	9	42
g++ -g <1>; a.out; g++ -g <1>; a.out; g++ -g <1>	28	1.00	2	217	1	388
<1>; vi <1>	464	0.99	3	6	4	4
vi <1>; gcc <1>; a.out; vi <1>; gcc <1>	25	1.00	4	257	7	469
<1>; vi <1>; gcc <1>; a.out	62	0.69	5	84	3	59
<1>; ls; more <1>	177	1.00	6	23	8	36
<1>; fg <1>; fg <1>; fg <1>; fg <1>; fg <1>	12	1.00	7	569	14	1397
<1>; gcc <1>; vi <1>; gcc <1>; vi <1>; gcc <1>	12	1.00	8	569	6	1397
vi <1>; gcc <1>; a.out; more <1>; vi <1>	17	1.00	9	393	3	785
f <1>; f <1>	120	0.98	10	40	18	76

Table 9: Normalized-Clustered Support/Confidence: Top 10 Sequences

Sequence	Freq	Conf	NC-SC pos	SC pos	NC-DD pos	DD pos
g++ -g <1>; a.out; g++ -g <1>; a.out; g++ -g	28	1.00	2	217	1	78
<1>; ls; cd <1>; ls	201	0.77	1	17	2	17
vi <1>; gcc <1>; a.out	92	0.67	5	55	3	31
<1>; vi <1>	464	0.99	3	6	4	4
cd <1>; ls; cd	219	0.36	1	15	5	26
<1>; gcc <1>; vi <1>; gcc	31	0.82	7	187	6	100
<1>; gcc <1>; a.out; vi <1>; gcc	25	0.69	4	257	7	144
<1>; ls; more <1>	177	1.00	6	23	8	36
cd <1>; ls; cd <1>; ls; cd <1>; ls; cd	29	0.59	1	208	9	249
f <1> | more	64	0.96	19	82	10	51

Table 10: Normalized-Clustered DD: Top 10 Sequences

Sequence	Description
ps -aux | grep <1>; kill -9	A user looking for a certain process id to kill.
tar <3>; cd; uuencode <2> > <1>; mailx -s <2> <	A user packaging a directory tree, encoding it to a file, and sending it by mail.
compress <1>;quota;compress <1>; quota	A user trying to overcome quota problems by compressing files.
latex <1>; dvips <1>; ghostview<1>; latex <1>; dvips <1>; ghostview<1>	A latex $write \rightarrow compile \rightarrow view$ cycle.
grep <2> | wc -l; grep <2> | wc	A user counting the number of occurrences of a certain expression in one or more files.
zcat <1> | tar xvf	A user uncompressing a zipped archive file and extracting files from it.
| awk <1> | sort -n | tail	A user processing data, sorting it arithmetically and examining the last part of it.

Table 11: Normalized-Clustered techniques: Samples of significant sequential patterns.

longer and more complex, user patterns than the basic algorithms did. For instance, the second pattern in Table 9, *g++ -g <1>; a.out; g++ -g <1>; a.out; g++ -g <1>*, reflects a $compile \rightarrow run$ cycle. The fourth pattern in the table, *vi <1>; gcc <1>; a.out; vi <1>; gcc <1>* represents an even more interesting $edit \rightarrow compile \rightarrow run$ cycle. The seventh sequence in Table 9, *<1>; fg <1>; fg <1>; fg <1>; fg <1>; fg <1>*, points at a repetitive behavior, which is quite typical for Unix users. Similarly, the ninth sequence of Table 10, *cd <1>; ls; cd <1>; ls; cd <1>; ls; cd*, reflects a typical repetitive behavior of a user traversing through several directories while examining the content of each of them.

In general, the results we received throughout the nine user datasets were similar, further demonstrating the ability of the improved techniques to discover valuable sequential patterns, which characterize interesting user behavior, and are overlooked by the basic methods. Additional examples of such user patterns from the other 8 datasets appear in Table 11.

6 Conclusions and Future Work

This paper tackles challenges in improving the quality of sequential pattern learning. We empirically compared several popular learning methods, to determine their relative strengths. Based on the comparison, we noted several common deficiencies in all tested algorithms: All are susceptible to a bias in preferring pattern candidates based on length; and all fail to generalize patterns, often taking a high-ranked pattern candidate as distinct from its shorter sub-patterns.

We presented techniques that are able to tackle these statistical biases that underlie many existing learning methods, in a general fashion. In that, the techniques we presented are method-neutral: They significantly improve all evaluated methods, as empirically demonstrated in thousands of trials on synthetic and real-world data.

We use a normalization method to effectively neutralize the length bias in all learning methods tested, by normalizing the frequency/significance rankings produced by the learning methods. Use of this method had improved accuracy by up to 42% in testing on synthetic data. We then use a clustering approach, based on a modified weighted edit-distance measure, to group together all patterns that are closely related. The use of clustering in addition to normalization had further improved accuracy by up to 22% in some cases. We also show that the techniques are robust to noise in and out of the patterns. Finally, the improved methods were run on two additional sets of data: sequences from Orwell's 1984, and UNIX real-world command-line data. The methods successfully detected many interesting patterns in both.

We believe that the implications of our work go beyond these results, in that they demonstrate that existing techniques may suffer from subtle biases, that must be tackled in a principled manner. However, a weakness with the methods that we presented is their use with very large data-bases. For instance, normalization requires counting all the patterns in the database, and would therefore be inefficient for large data-mining applications. However, the techniques are well suited for typical agent-observation data (such as the UNIX command-line data). We plan to consider large data-mining applications in our future work.

Acknowledgments

A subset of preliminary results from this work appears in [9]. The authors gratefully acknowledge useful discussions with Adele Howe, Paul Cohen, and Ronen Feldman. Special thanks to K. Ushi.

References

[1] C. C. Aggarwal and P. S. Yu. A new framework for itemset generation. pages 18–24, 1998.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[4] M. Bauer. From interaction data to plan libraries: A clustering approach. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99)*, volume 2, pages 962–967, Stockholm, Sweden, August 1999. Morgan-Kaufman Publishers, Inc.

[5] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In J. Peckham, editor, *SIGMOD 1997,*

Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA, pages 255–264. ACM Press, 05 1997.

[6] P. Cohen and N. Adams. An algorithm for segmenting categorical time series into meaningful episodes. *Lecture Notes in Computer Science*, 2189, 2001.

[7] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 67–76. ACM Press, 2001.

[8] S. Hettich and S. D. Bay. The uci kdd archive. http://kdd.ics.uci.edu/, 1999.

[9] Y. Horman and G. A. Kaminka. Removing statistical biases in unsupervised sequence learning. In *Proceedings of Intelligent Data Analysis (IDA-05)*, Madrid, Spain, 2005.

[10] A. E. Howe and P. R. Cohen. Understanding planner behavior. *Artificial Intelligence*, 76(1–2):125–166, 1995.

[11] G. A. Kaminka, M. Fidanboylu, A. Chang, and M. Veloso. Learning the sequential behavior of teams from observations. In *Proceedings of the 2002 RoboCup Symposium*, 2002.

[12] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In N. R. Adam, B. K. Bhargava, and Y. Yesha, editors, *Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401–407. ACM Press, 1994.

[13] T. Lane and C. E. Brodley. Temporal sequence learning and data reduction for anomaly detection. *ACM Transactions on Information and System Security*, 2(3):295–331, 1999.

[14] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, 1997.

[15] T. Oates and P. R. Cohen. Searching for structure in multiple streams of data. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 346–354, 1996.

[16] J. S. Park, M.-S. Chen, and P. S. Yu. Using a hash-based method with transaction trimming for mining association rules. *Knowledge and Data Engineering*, 9(5):813–825, 1997.

[17] P. Sellers. The theory and computation of evolutionary distances: pattern recognition. *Journal of Algorithms*, (1):1:359–373, 1980.

[18] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1):39–68, 1998.

[19] R. R. Sokal and F. J. Rohlf. *Biometry: The Principles and Practice of Statistics in Biological Research*. W.H. Freeman and Co.„ New York, 1981.

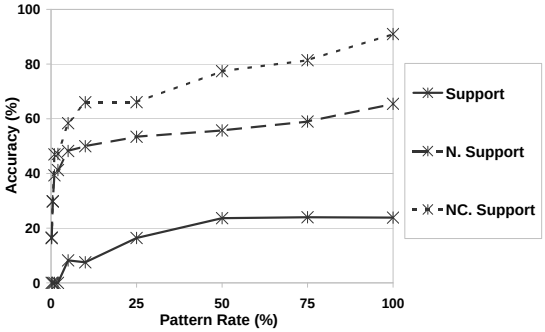
[20] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. M. G. Apers, M. Bouzeghoub, and G. Gardarin, editors, *Proc. 5th Int. Conf. Extending Database Technology, EDBT*, volume 1057, pages 3–17. Springer-Verlag, 25–29 1996.

[21] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–41. ACM Press, 2002.

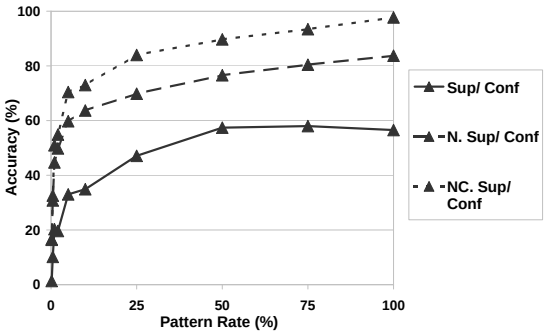
[22] M. Zaki. Fast mining of sequential patterns in very large databases. Technical Report 668, University of Rochester Compute Science Department, 1997.

[23] M. Zaki, N. Lesh, and M. Ogihara. Planmine: Sequence mining for plan failures. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 369–374. AAAI Press, 1998.

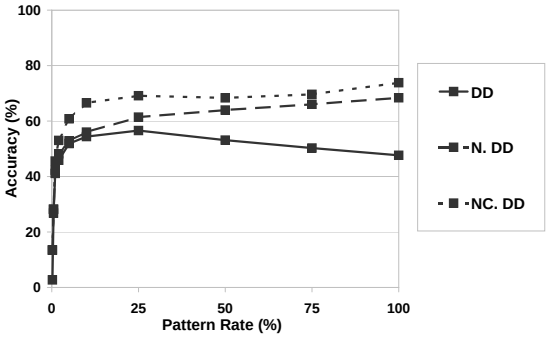
[24] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. Technical Report TR651, 1997.



(a) Support

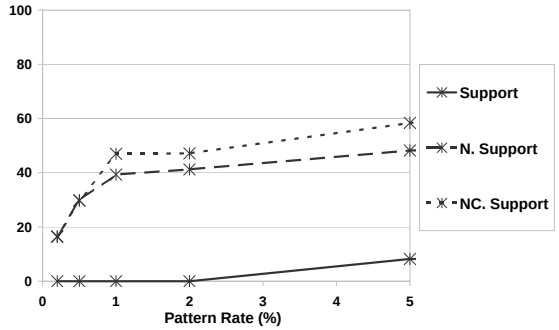


(b) Support/Confidence

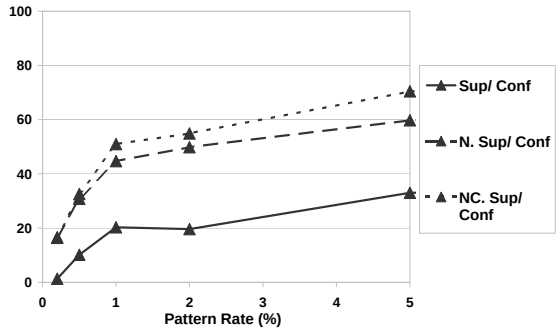


(c) DD

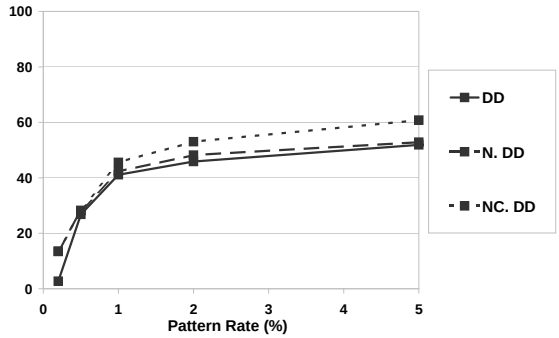
Figure 6: Modified: All Pattern Rates, average results.



(a) Support

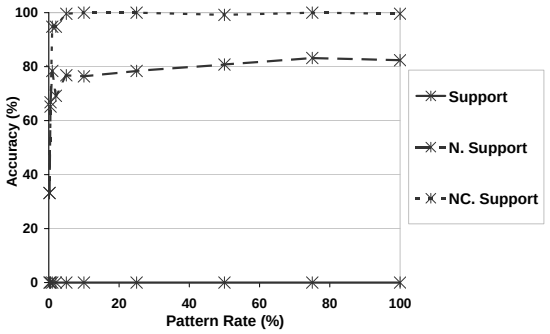


(b) Support/Confidence

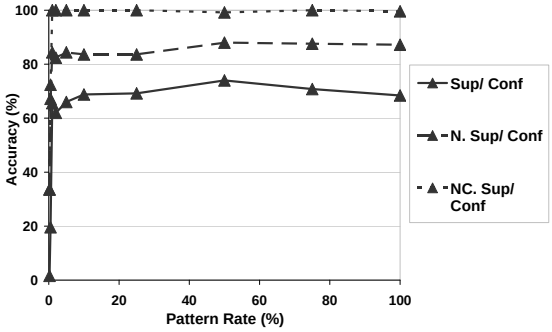


(c) DD

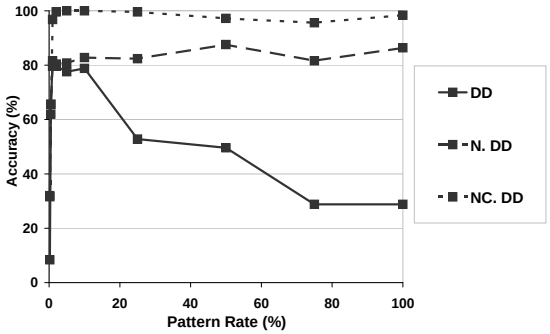
Figure 7: Modified: Low Pattern Rates, average results.



(a) Support

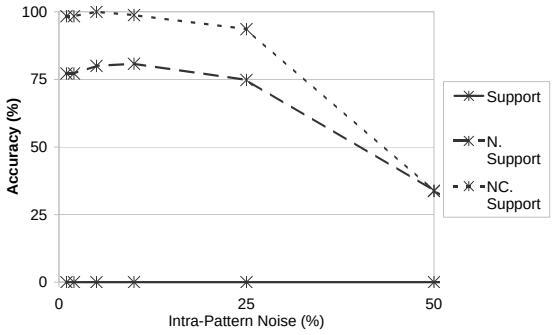


(b) Support/Confidence

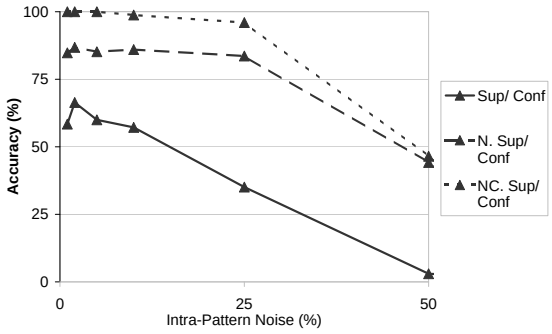


(c) DD

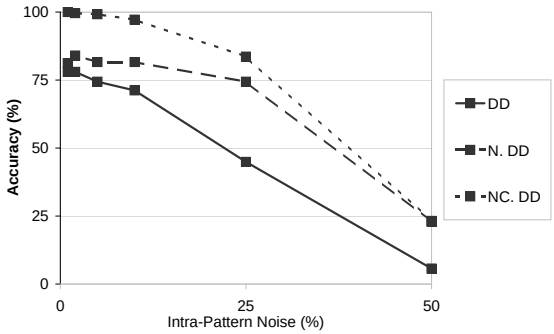
Figure 8: Modified: $T = 26$, Patterns of size 3–5.



(a) Support



(b) Support/Confidence



(c) DD

Figure 9: Handling intra-pattern noise.

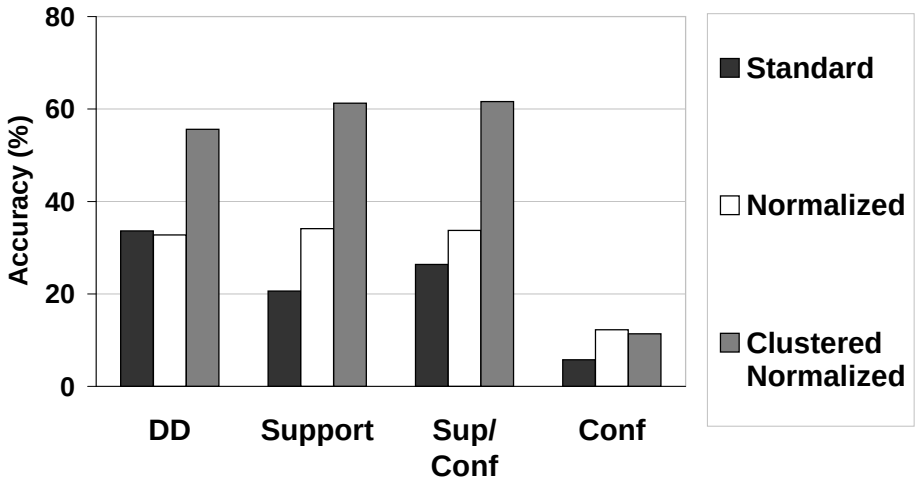


Figure 10: **Accuracy improvements: Orwell's 1984.**