Mach Learn (2009) 75: 91–127
DOI 10.1007/s10994-008-5098-y

Mining probabilistic automata: a statistical view of sequential pattern mining

Stéphanie Jacquemont · François Jacquenet ·
Marc Sebban

Received: 9 March 2007 / Revised: 12 December 2008 / Accepted: 16 December 2008 / Published online: 8 January 2009
Springer Science+Business Media, LLC 2009

Abstract During the past decade, sequential pattern mining has been the core of numerous research efforts. It is now possible to efficiently extract knowledge of users' behavior from a huge set of sequences collected over time. This has applications in various domains such as purchases in supermarkets, Web site visits, etc. However, sequence mining algorithms do little to control the risks of extracting *false discoveries* or overlooking *true knowledge*. In this paper, the theoretical conditions to achieve a relevant sequence mining process are examined. Then, the article offers a statistical view of sequence mining which has the following advantages: First, it uses a compact and generalized representation of the original sequences in the form of a probabilistic automaton. Second, it integrates statistical constraints to guarantee the extraction of significant patterns. Finally, it provides an interesting solution in a privacy preserving context in order to respect individuals' information. An application in car flow modeling is presented, showing the ability of our algorithm (ACSM) to discover frequent routes without any private information. Comparisons with a classical sequence mining algorithm (SPAM) are made, showing the effectiveness of our approach.

Keywords Sequence mining · Probabilistic automaton · Privacy preserving data mining

1 Introduction

Sequential pattern mining aims to automatically find subsequences (also called patterns) that appear *frequently* (*i.e.* more than a support threshold) in a set of sequences. Many

Editors: Thomas Gärtner, Gemma C. Garriga.

S. Jacquemont (✉) · F. Jacquenet · M. Sebban
Laboratoire Hubert Curien, UMR 5516 Université Jean Monnet, 18, rue Benoît Lauras, Bâtiment F, 42000 Saint-Étienne, France
e-mail: jacqstep@univ-st-etienne.fr

F. Jacquenet
e-mail: jacquene@univ-st-etienne.fr

M. Sebban
e-mail: sebbanma@univ-st-etienne.fr

algorithms have been proposed during the last decade (Mannila et al. 1997; Zaki 2000; Garofalakis et al. 2002; Pei et al. 2002). Completeness and correctness are two main features that must be satisfied by those algorithms. They actually have to extract all the frequent patterns (*completeness*) and only these ones (*correctness*). There have been a lot of applications based on such techniques (see Han et al. 2002 for a survey). The first one, proposed by Agrawal and Srikant (1995), concerned the discovery of customers' behavior in supermarkets over time in the form of subsequences of purchases. That knowledge may be very valuable for the marketing departments of companies. In the domain of manufacturing supervision, sequence mining can be used to discover frequent patterns of alarms. It may help supervisors while searching for defaults in a plant (Klemettinen et al. 1999). More recently, in the domain of Web Mining (Kosala and Blockeel 2000), various systems have been designed to make use of Web logs in order to model the behavior of Web users (Spiliopoulou and Pohle 2001). In this paper, we present another possible application that consists of discovering routes frequently used by car drivers in a town.

From a statistical point of view, a sequence mining process is always achieved on a finite sample set *LS* of sequences that has been drawn from an unknown target distribution. For example, *LS* can be a set of English sentences, and the target distribution is the whole English language of infinite size. It is important to note that in some domains, it can be difficult to collect a large dataset *LS*. The data acquisition could actually be expensive (in time or money) to manage (that can be the case in molecular biology for example), or simply impossible due to the limited number of cases that can be observed in the real world (for instance the number of breakdowns of a system). In those cases, sequence mining algorithms are almost always applied without any consideration of the underlying statistical distribution from which *LS* has been drawn. However, the smaller *LS*, the higher the risk to have a large statistical bias. In other words, the algorithms make no assessment of the likelihood that an extracted pattern is an artifact of the sampling rather than a consistent pattern in the target distribution. The **first contribution** of our paper consists of taking into consideration this problem by proposing a theoretical framework to verify under which constraints a sequence mining process is statistically relevant. We provide a lower bound on the number of sequences needed to guarantee the discovery of significant knowledge. However, what happens when this bound is not satisfied in a given application? To answer this question, we examine in our **second contribution** a statistical view of sequence mining. Our motivation is based on the following remark: if the number of sequences of *LS* is insufficient to satisfy the lower bound, we can try to generalize them in the form of a probabilistic generative model. In other words, such a generative model will not only be able to cover the examples of *LS* but also it will enable other sequences to be represented. Probabilistic Deterministic Finite state Automata (PDFA) and Hidden Markov Models are such probabilistic models that can be used to generalize the sequences of *LS*. It can be theoretically shown (Dupont et al. 2005) that under certain conditions the two models are equivalent. In this article, we choose to work on PDFA and we set the theoretical conditions to learn a good automaton to be used in a sequence mining process.

Moreover, to take into account a current trend in sequence mining, we introduce *constraints* on the extracted frequent sequences. Actually, despite the use of a support threshold, an *unconstrained* search can produce millions of patterns or may even be intractable. A recent strategy consists of extracting frequent patterns under constraints such as length and width restrictions, minimum or maximum gap between events, time window of occurrence (Zaki 2000), or regular expressions (Garofalakis et al. 2002) (see also Pei et al. 2002). Moreover, as Zaki claims in (Zaki 2000), there exist many domains (such as bioinformatics) where the user may be interested in interactively adding syntactic constraints on the mined sequences. In this paper, we introduce two new constraints that we adapt to the specific context of probabilistic automata. The first one consists of extracting only frequent patterns which begin after prefixes of a given length. The second constraint is based on the assessment of the statistical significance of the extracted frequent patterns. This constitutes the core of our new paradigm of statistical sequence mining.

The numerous applications prove that sequence mining algorithms are useful tools for discovering knowledge in various situations. Nevertheless, if the goal is to discover knowledge from the observation of human behavior, we claim that these algorithms do not preserve individuals' privacy. For example, regarding the problem of discovering frequent sequences of Web pages visited over time by users of a web site, the input data are, at least, the IP address of the visitors and the Web pages they have browsed. This is a great breach to their privacy and users might want their private information not to be logged on the servers. Considering another example, to study frequent routes in a town one could install many Web cams and trace the route for each driver. This is obviously a nonacceptable breach to the privacy of car drivers.

In this context, privacy preserving data mining has become an important subject of research with a lot of possible real world applications. There is actually a great demand from users, and more generally from society, that data miners preserve their privacy. For example, considering the data generated by surveillance of frequent routes traversed by car drivers, we may think about de-identifying images from Web cams (Newton et al. 2005); but this is a costly solution. Moreover, installing a Web cam in every street of a town may lead to opposition from people. Our **third contribution**, in this paper, is to show how PDFA-based sequence mining provides a less costly solution for privacy preserving problems that may be stated as flow control problems.

The rest of this paper is organized as follows. In Sect. 2, after having presented some recent approaches that deal with the significance of the extracted patterns, we present a lower bound on the number of sequences needed for a relevant sequence mining task. Section 3 introduces a statistical sequence mining framework by describing the way we can replace the original set of sequences by a probabilistic automaton. Section 4 presents a theoretical and experimental study on the conditions an inference algorithm must fulfill to guarantee the construction of a good automaton that is useful for statistical sequence mining. Section 5 shows how to integrate new constraints in a PDFA-based sequence mining algorithm called ACSM. Section 6 explains how using an automaton can offer a good solution for privacy preserving data mining. Finally, we present a traffic simulator able to discover frequent routes in a town. Besides the description of the system, we experimentally compare our algorithm with a classical sequence mining algorithm.

2 Lower bound on the number of sequences

In this section, before detailing our lower bound, we present some related works dealing with the significance of the extracted patterns.

2.1 Related work

In a sequence mining process, one extracts frequent patterns from a set LS of N sequences. To achieve this task, one compares, for each tested pattern w, its observed frequency $\hat{p}(w)$ to a given support threshold p_0. The following decision rule is then applied: if $\hat{p}(w) > p_0$, w is considered as frequent. As we mentioned in the introduction, the extracted patterns represent knowledge that may be very valuable in many applications. However, the exploitation without any risk of this knowledge requires the assumption that *LS* has been correctly drawn from an unknown underlying distribution. Although classical sequence mining algorithms, such as SPAM (Ayres et al. 2002), SPADE (Zaki 2001) or GSP (Srikant and Agrawal 1996), are correct and complete on *LS*, there is no guarantee that the extracted patterns actually describe relevant information in the unknown distribution from which *LS* has been drawn. In other words, some patterns may have been extracted due to chance alone (we will call them *false positives*), while some true frequent patterns may have been overlooked because of the sampling (called *false negatives*).

Let us describe in the following two main solutions that have been presented in the literature to control the significance of the extracted patterns.

2.1.1 Use of Chernoff bounds

Rather than directly comparing the observed frequency $\hat{p}$ in *LS* with p_0, a first solution consists of bounding p_0 in order to take into account the estimate error $|\hat{p} - p|$ due to the use of a sample *LS* of finite size, where p is the true probability of the pattern under the unknown theoretical distribution.

A well-known nonparametric approach that deals with this problem is based on Chernoff bounds that state that the estimate error between a random variable X observed on a sample *LS* and its expected value $E(X)$ according to a target distribution is lower bounded by ϵ, such that

$$\forall \epsilon \in]0,1[, \quad P(|X - E(X)| \geq \epsilon) < e^{-2N\epsilon^2}. \tag{1}$$

Let the observed frequency $\hat{p}$ be the random variable X of inequality (1) computed from *LS*, and p its expected value $E(X)$ over the theoretical distribution, the Chernoff bounds can be rewritten as follows:

$$P(|\hat{p} - p| \geq \epsilon) < e^{-2N\epsilon^2}. \tag{2}$$

Inequality (2) can be used in different ways. First, given a size N, solving for ϵ this inequality equal to a given probability provides a slack value of the support threshold p_0. On the other hand, given a value ϵ, solving for N inequality (2) equal to δ provides a lower bound of the sample size N satisfying the estimate error ϵ. Despite its obvious advantages, the use of the Chernoff bounds has a limitation. Indeed, the symmetry due to the absolute value in inequality (2) indicates that the risk of a bad estimation $\hat{p}$ is equally distributed around p. In other words, the risk that a pattern occurs in *LS* less often than expected in the theoretical distribution is equal to the risk that a pattern occurs more often than expected. In this context, Chernoff bounds does not allow the distinction between the false positive rate and the false negative one. This can be a problem in domains where both risks have to be independently handled.

Recently, Laur et al. proposed in (Laur et al. 2007a, 2007b) an approach that not only makes use of Chernoff bounds but also deals with both risks. The authors show how to only extract true frequent patterns with a high probability while controlling the false negative rate. Inversely, they show how to guarantee the extraction of all the true frequent patterns with a high probability while limiting the false positive rate. Even if this approach is theoretically well founded, the user has to choose the criterion he wants to optimize (the false positive or false negative rate), that can be a tricky task in domains where both risk are definitely undesirable.

2.1.2 Use of statistical tests

A second solution to check the relevance of a discovered pattern is to resort to statistical tests. In this framework of statistical inference, one have to test a so-called null hypothesis H_0 against an alternativeone H_a. The standard approach in sequence mining consists of defining H_0 as the not interesting situation, and H_a as the situation of an important discovery. Let us recall that when a test is performed, there are two kinds of possible risks of errors, usually called α and β, as described in Table 1 (Megiddo and Srikant 1998). α represents the risk of rejecting a correct null hypothesis and β is the risk of not rejecting a false null hypothesis. Therefore, in our framework, α corresponds to the false positive rate, and β is the false negative rate.

Megiddo and Srikant (1998) deal with the evaluation of the quality of association rules extracted from a set of N transactions by using such a statistical test-based approach. They consider association rules $X \Rightarrow Y$, where X and Y are sets of items. As a null hypothesis, they assume that X and Y occur in transactions independently. Thus, they test the null hypothesis $H_0 : p(X \cap Y) = p(X) \times p(Y)$ against the alternative one $H_a : p(X \cap Y) > p(X) \times p(Y)$, which, roughly speaking, means that a lot of transactions that contain X also contain Y. They perform this statistical test exploiting the property that the observed frequency of an itemset asymptotically follows a normal distribution. To reduce the risk of accepting a false discovery (*i.e.* to reduce α), they increase the support threshold p_0 by $z_\alpha \times \sigma_{\hat{p}}$, where $\sigma_{\hat{p}}$ is the standard deviation of $\hat{p}$ and z_α is the $(1-\alpha)$ percentile of the normal distribution. Therefore, by a priori tuning α, they can control the false positive rate. Nevertheless, using a small value for α results in the increase of the false negative rate β.

Recently, Webb (2007) presents two new approaches to applying statistical tests in pattern discovery to assess the quality of a pattern. First, he suggests the splitting of the set of sequences *LS* into an *exploratory* set, from which a pattern extraction is achieved, and a *holdout* set used to assess the quality of each pattern. Despite promising experimental results, Webb does not provide any bound that enables both risks α and β to be reduced. He also presents an approach based on the Bonferroni adjustment (Shaffer 1995) to take into account the multiple testing issues. When a statistical test is applied many times during an assessment, a special problem actually arises: for instance, if α corresponds to the risk of taking a wrong single decision, repeating the test many times globally increases that risk (Shaffer 1995). To overcome this drawback, several strategies have been proposed. A famous one is the Bonferroni adjustment that uses a risk α/n when performing n hypothesis tests. However, if n is large, such adjustment turns out to be strict and leads to the increase of the other risk β.

Another solution consists of using Holm procedure (Holm 1979) that takes into account the *p-value* of each test and orders them to tune a less strict risk. Such a strategy is also used in the BH procedure (Benjamini and Hochberg 1995) that aims to fix α while controlling β. However, both of these adjustments require the computation of the *p-values* of the n tests which depend on the current application.

Table 1 Risks of errors while performing a statistical test

		Truth	
		H_0 true	H_a true
Decision	Not reject H_0	Correct decision	β
	Reject H_0	α	Correct decision

In the next section, we aim to provide a more general tool whatever the application we deal with. We aim to define a relationship between the number N of sequences to mine by a sequence mining algorithm and a priori fixed risks α and β. Therefore, we assume that we do not yet have the sequences themselves before computing our bound. Note that the theorem we present holds for any single pattern, but since our bound is a function of α and β, it is obviously possible to apply strategies such as the Bonferroni adjustment to overcome multiple testing issues. So, even if the multiple testing issues remain present, we can consider that they are not directly in the scope of our proposed approach.

Before ending this section, note that other investigations have dealt with the assessment of the statistical significance of patterns in data mining. They often use efficient statistical tests (such as the Chi-square test and Fisher exact test) to statistically measure the level of dependency between the components of a pattern. An often used strategy consists of verifying if the extracted structure would also be discovered from other samples having same margins (see Gionis et al. 2006 for example). Note that such nonparametric approaches are interesting because they do not impose a condition on the nature of the underlying distribution. This can be useful when the approximation conditions (to the normal distribution, for instance) are not fulfilled.

2.2 Lower bound on N

As defined before, $\hat{p}(w)$ is the proportion of sequences in the set LS that contain a pattern $w = \langle x_1 \cdots x_l \rangle$ where $x_1, \ldots, x_l$ are (possibly nonconsecutive) symbols in the original sequences of LS. For instance, $w = AGT$ is a pattern which occurs twice in $LS = \{ACGAT, CAGCT, AAG\}$. Therefore, in this case, $\hat{p}(AGT)$ is equal to $\frac{2}{3}$ and has to be compared to a support threshold p_0; if $\hat{p}(w) > p_0$, w is then considered as frequent, otherwise it is considered as nonfrequent. In this statistical context, $\hat{p}(w)$ is nothing else but an estimate of the real probability $p(w)$ (note that $p(w)$ depends on the length of the sequences which is also determined by the target distribution). Since $p(w)$ is unknown, one can formulate a hypothesis on its real value and achieve a statistical test. As usually done in the standard approaches, we suggest to describe by the null hypothesis H_0 the situation where $\hat{p}(w)$ is not high enough to consider w as being frequent. As done in (Megiddo and Srikant 1998), we suggest to keep the maximal value p_0 that prevents w from being accepted as frequent. Therefore, we test the null hypothesis H_0: $p(w) = p_0$, against the alternative one H_a, which describes an interesting discovery, *i.e.* H_a: $p(w) > p_0$.

Risk of errors α In our sequence mining context, α corresponds to the false positive rate, or in other words, the probability to *accept a false frequent pattern*. For instance, with a support threshold p_0 of 10%, observing $\hat{p}(w) = 10.2\%$ in LS does not mean that w is definitely frequent in the target distribution. To be able to take a well-founded decision, we can a priori fix α (usually 5%, but it can depend on the application we deal with), and then compute a bound of rejection k, satisfying α. More formally,

$$\alpha = P\left(\hat{p}(w) > k | H_0 \text{ true}\right). \tag{3}$$

The number of sequences of LS that contain w is a binomial random variable with success probability $p(w)$. According to the number N of sequences and the support threshold p_0, we can use either the normal approximation or the Poisson approximation. In our context, we aim to provide a theoretical bound on N that will be by nature quite large. Moreover, since we are looking for frequent patterns, we can assume that p_0 will be chosen to be sufficiently large allowing us to use the normal approximation. Even if the use of the exact binomial distribution or the Poisson approximation would also enable us to get a bound, this would complexify the computation of that bound due to the discrete nature of these distributions. Therefore, using the central limit theorem, we will consider in the following that the proportion $\hat{p}(w)$ follows a normal distribution $\mathcal{N}$, such that

$$\hat{p}(w) \approx \mathcal{N}\left(p(w), \sqrt{\frac{p(w)(1-p(w))}{N}}\right).$$

Equation (3) can be rewritten

$$\alpha = P\left(\frac{\hat{p}(w) - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} > \frac{k - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} | H_0 \text{ true}\right). \tag{4}$$

Since H_0 is true, we have to replace $p(w)$ by its value under H_0. We get

$$\alpha = P\left(\frac{\hat{p}(w) - p_0}{\sqrt{\frac{p_0(1-p_0)}{N}}} > \frac{k - p_0}{\sqrt{\frac{p_0(1-p_0)}{N}}}\right). \tag{5}$$

We can then easily deduce the bound k which corresponds to the $(1-\alpha)$-percentile z_α of the normal distribution:

$$k = p_0 + z_\alpha \sqrt{\frac{p_0(1-p_0)}{N}}. \tag{6}$$

To recap, by fixing a risk α, (6) gives us the bound of rejection of H_0. For example, let us suppose we are mining $N = 10000$ sequences. We fix the support threshold $p_0 = 10\%$ and a risk $\alpha = 5\%$ ($z_\alpha = 1.645$ by reading the table of the normal distribution). Plugging these values in (6), we get $k = 0.1 + 1.645 \times \sqrt{\frac{0.1*0.9}{10000}} = 0.105$. Therefore, a pattern w with a support $\hat{p}(w) = 10.2\%$ will be in fact rejected in order to control the risk α of accepting false positives.

Risk of errors β Regarding β, it describes the probability to *reject a true frequent pattern*. In contrast to α, β can be calculated according to the computed bound k. Since H_a: $p(w) > p_0$ is true, we have to set a given value for $p(w)$ satisfying the constraint $p(w) > p_0$. Let p_a be this value (in practice, a threshold close to p_0). We get

$$\beta = P(\hat{p}(w) < k | H_a \text{ true}). \tag{7}$$

As previously done for α,

$$\beta = P\left(\frac{\hat{p}(w) - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} < \frac{k - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} | H_a \text{ true}\right). \tag{8}$$

By replacing $p(w)$ by its value under H_a, we get

$$\beta = P\left(\frac{\hat{p}(w) - p_a}{\sqrt{\frac{p_a(1-p_a)}{N}}} < \frac{k - p_a}{\sqrt{\frac{p_a(1-p_a)}{N}}}\right). \tag{9}$$

Since k is known thanks to (6), the $(1-\beta)$-percentile z_β is also known, and β can be easily deduced from the normal distribution. To continue with our previous example (assuming that $N = 10000$ sequences), let us suppose that $p_a = 11\%$, then $\beta = 5.5\%$ (by reading the table of the normal distribution). Therefore, for a true support of 11%, the risk to falsely accept the null hypothesis based on a finite sample of $N = 10000$ sequences is 5.5%.

The objective of a relevant sequence mining process is to reduce both risks α and β. However, there exists a trade-off between them. Given a fixed number of sequences N, β increases if one reduces α and vice versa. On the other hand, one can wonder how many sequences are needed to not exceed a priori fixed α and β risks. We define here a lower bound on N that answers this question.

Theorem 1 *Let w be a pattern and LS a set of sequences independently drawn from a target distribution $\mathcal{D}$ on which a sequence mining algorithm $\mathcal{A}$ is run. Let $p_0 \in [0,1]$ be the support threshold used by $\mathcal{A}$. Given α and β, two fixed parameters respectively corresponding to the risk of extracting w from LS with $\mathcal{A}$ as being a false positive (resp. false negative) pattern.*

To satisfy α and β, the minimal size N_{low} of LS must be equal to

$$N_{low} = \left[\frac{z_\beta\sqrt{p_a(1-p_a)} + z_\alpha\sqrt{p_0(1-p_0)}}{p_a - p_0}\right]^2, \quad 0 < p_0 < p_a < 1,$$

where z_α (resp. z_β) is the $(1-\alpha)$ percentile (resp. $(1-\beta)$ percentile) of the normal distribution and where $p_a \in]p_0, 1]$ is the support threshold considered by the user for describing a false negative pattern.

Proof The proof is straightforward. We can deduce from (9) that

$$k = p_a - z_\beta\sqrt{\frac{p_a(1-p_a)}{N}}. \tag{10}$$

Equating (6) to (10), we can deduce that

$$p_0 + z_\alpha\sqrt{\frac{p_0(1-p_0)}{N}} = p_a - z_\beta\sqrt{\frac{p_a(1-p_a)}{N}}. \tag{11}$$

Extracting N from (11), we obtain the lower bound. □

Let us now describe the meaning of this bound. It is important to note that there is a direct relationship between β and p_a given a fixed number of sequences. Indeed, as described in Fig. 1, p_a is the expectation of $\hat{p}(w)$ under the alternative hypothesis H_a. β corresponds to the density of the normal distribution beneath the bound k of rejection of H_0. Therefore, the farther p_a is from p_0, the lower the risk β. Since β and p_a are parameters in our lower bound, reducing both implies an increase of the needed number of sequences. The same remark can be done between α and β. Reducing α for a given size N implies the increase of β. Therefore, reducing both risks results in the increase of the required number of sequences.

To illustrate this lower bound, the chart of Fig. 2 shows the evolution of N_{low} according to α, β, p_0 and p_a. For the sake of legibility we choose $\alpha = \beta$. We plot two curves with two different values of p_a. We can note that the smaller $p_a - p_0$, the larger the lower bound.

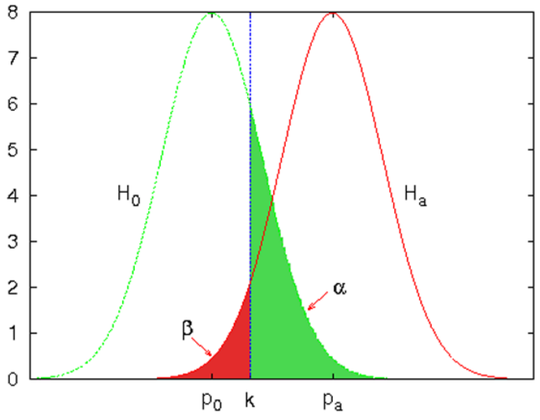


Fig. 1 Trade-off between α (*light grey area*) and β (*dark grey area*). p_0 (resp. p_a) is the expectation of $\hat{p}(w)$ under H_0 (resp. H_a)

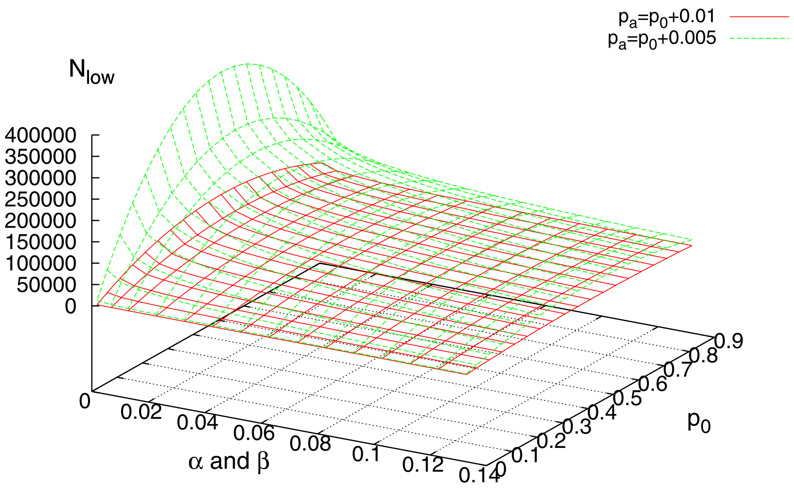


Fig. 2 N_{low} according to α, β, p_0 and p_a

2.3 Illustration of our bound on a real and on an artificial example

In this section, we first illustrate the impact of our bound on a real world application. We carried out a series of experiments on the ATIS (Air Travel Information Service) corpus. This database consists of information requests performed in English. We have an original set Ω of 14044 sentences from which we randomly draw samples LS_i of increasing size $|LS_i|$ (from 10 to 14044) and we extract frequent patterns with a support threshold p_0 of 10% with SPAM (Ayres et al. 2002). In this series of experiments, we assume that Ω represents the underlying distribution of the samples LS_i that we aim to estimate (usually, let us recall that this one is unknown). In order to assess the effect of the size $|LS_i|$ on the quality of this estimate, we have to be able to measure the empirical risks of α and β, that we will call $\hat{\alpha}$ and $\hat{\beta}$. $\hat{\alpha}$ is the observed proportion of patterns that have been extracted as frequent from LS_i while they are not frequent in the target population Ω. $\hat{\beta}$ corresponds to the observed proportion of patterns that are frequent in Ω but overlooked from LS_i.

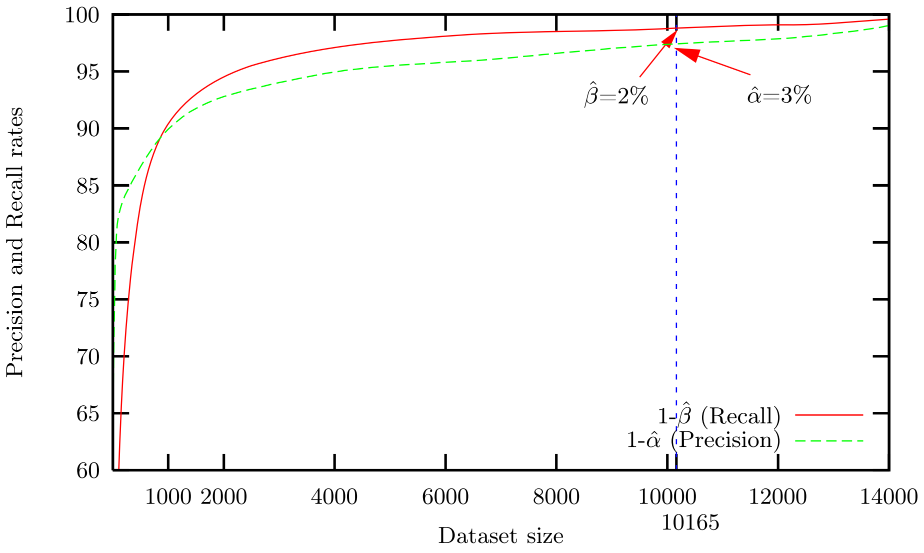


Fig. 3 Evolution of the quality of the results of a sequence mining algorithm according to an increasing size of LS_i

Figure 3 describes, according to an increasing size $|LS_i|$ of the sample set LS_i and a support threshold $p_0 = 10\%$, the evolution of $1 - \hat{\alpha}$, usually called the *precision*. Figure 3 also shows $1 - \hat{\beta}$, usually called the *recall*, that has been obtained using a value $p_a = 11\%$. Note that we performed 15 trials, for each size $|LS_i|$, and we computed the average in order to reduce the variance of the results. As expected, the higher the number of sequences, the smaller the computed risks $\hat{\alpha}$ and $\hat{\beta}$. We can also note that for small sizes of LS_i (<1000) both risks $\hat{\alpha}$ and $\hat{\beta}$ are high (>10%) meaning that a lot of extracted patterns are not truly frequent in Ω and many others have been overlooked. This example is a good illustration of the bottleneck of standard sequence mining algorithms.

Since $\hat{\alpha}$ and $\hat{\beta}$ can be empirically measured, they can be compared with the theoretical risks α and β to verify the relevance of our bound. To achieve this task, let us compute N_{low} for given theoretical parameters α, β, p_a and p_0. For instance, let us set $\alpha = \beta = 5\%$ ($p_a = 10\%$ and $p_a = 11\%$ being already fixed). Plugging these values in our bound yields the value $N_{low} = 10165$. If we observe from Fig. 3 the results obtained from 10165 sequences, we can conclude that our bound is relevant because the two observed errors computed on the ATIS database ($\hat{\alpha} = 2\%$ and $\hat{\beta} = 3\%$) actually do not exceed our a priori fixed theoretical risks α and β.

Note that the difference between the observed and the theoretical rates can appear quite substantial on this experiment even if it is on the "safe side". In fact, the distance between the observed and the theoretical risks directly depends on the sample LS_i drawn from the unknown target distribution. But since N_{low} constitutes a lower bound needed, *in the worst case*, to satisfy α and β, our theorem states that we never fall on the "unsafe side". An interesting perspective of this work would consist of taking into account some information about the set LS (for example, the empirical distribution of the supports), to improve the assessment of the theoretical bound and then reduce its pessimism in some situations.

To illustrate the use of our bound in a situation where the theoretical distribution is a priori known, we used a simulated example in a second series of experiments. We generated

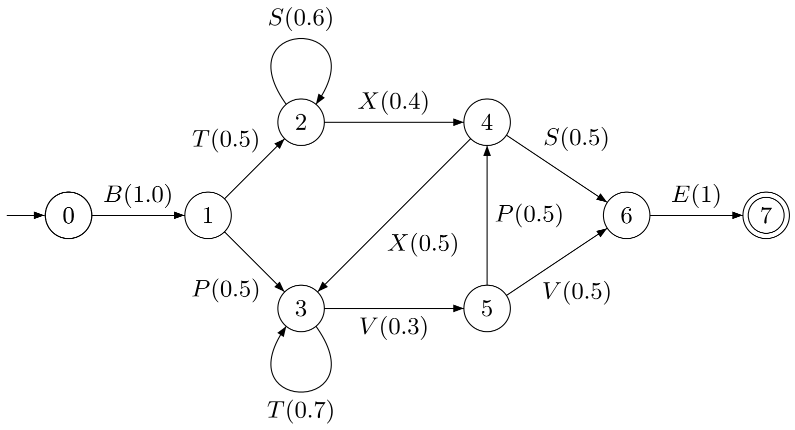


Fig. 4 Automaton corresponding to the Reber grammar

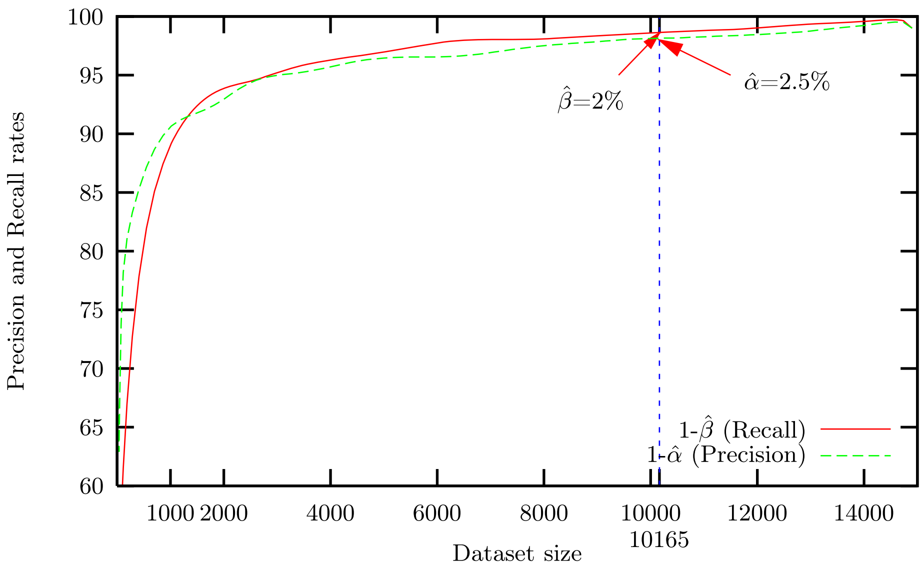


Fig. 5 Evolution of the quality of the patterns extracted from the Reber grammar

various databases from the Reber grammar (Reber 1967) whose target distribution is an automaton made up of 8 states and an alphabet of 7 letters (see Fig. 4).

Note that such an automaton constitutes a theoretical distribution from which it is possible to compute the true probability $p(w)$ of any pattern w, using suitable calculation methods (see the approach of Hingston 2002 for example, that will be detailed in the next part of this paper). As done before, we can compute for given parameters α, β, p_0 and p_a our lower bound N_{low} and compare it with the observed rates $\hat{\alpha}$ and $\hat{\beta}$. Using again $\alpha = 5\%$, $\beta = 5\%$, $p_0 = 10\%$ and $p_a = 11\%$, we obtain $N_{low} = 10165$ sequences. By observing the corresponding empirical risks on the curves of Fig. 5, we can note that they are smaller than 5%; therefore, once again, our theorem holds.

Despite its advantages, our lower bound might be difficult to reach in some domains. This can be the case in biological or medical applications. For example, DNA sequences are difficult and expensive to produce. Paradoxically, such domains are those which require rigorous decisions because the results concern a problem of public health. To overcome this drawback, one solution consists of generalizing the input sequences in the form of a generative model in order to cover more examples than those of *LS*. To achieve this task, we show in Sect. 3 that we can use Probabilistic Deterministic Finite state Automata (PDFA) that also have the advantage to constitute a compact representation of the data. We will investigate in Sect. 4 what is the statistical significance of a PDFA.

3 A graph-based statistical view of sequence mining

3.1 Related work

The use of a graph structure has already been exploited by Borges and Levene to extract frequent patterns. In (Borges and Levene 1998), they define the concept of *composite association rule* processed from a structured directed graph, built from the log files of a Web site. They propose two algorithms to find trails in the graph with confidence and support higher than given thresholds. In (Borges and Levene 1999), they use a more specific structure called *Hypertext Probabilistic Grammar*. They use *Ngrams* to reduce the history depth and find user navigation patterns. They show (Borges and Levene 2004) how to use higher-level Markov models in order to process a weighted automaton to discover frequent paths of Web site users from log files. They use state cloning to duplicate states such that the first-order probabilities induced by its out-links diverge significantly from the corresponding second-order probabilities. They also present a clustering algorithm to identify the best way to distribute a state's in-link between the state and its clones. Nevertheless, all these approaches only aim to discover sequential patterns made up of consecutive Web pages. Dupont et al. (2006) present a method to extract relevant subgraphs between nodes of interest with Hidden Markov Models. They use random walks to calculate those subgraphs by considering the frequency of use of each edge. Here again, the extracted patterns are connected components. To achieve a more general sequence mining task, an efficient tool should be able to discover patterns made up of (potentially) non-consecutive elements. Hingston (2002) proposes a first solution by modeling a set of sequences in the form of a probabilistic deterministic finite state automaton (PDFA), from which frequent patterns are extracted. Let us first recall some definitions about PDFAs before presenting Hingston's method, which constitutes the starting point of our approach.

3.2 Grammatical inference and PDFA

Modeling a sample of sequences by a PDFA can be viewed as a way to generalize this set. Learning such a PDFA is one of the aims of grammatical inference (de la Higuera 2005) which aims to infer a grammar from a set of examples (positive and negative, or positive only). In other words, the objective is to find the language from which the set of sequences has been generated. When the language is regular, it can be actually modeled in the form of a PDFA. More formally, a PDFA can be defined as follows:

Definition 1 A PDFA $A = \langle Q, \Sigma, q, q_0, \pi, \pi_F \rangle$ is a tuple where:

- Q is a finite set of states;
- Σ is the alphabet;

Table 2 Set of 15 sequences built from the alphabet $\Sigma = \{a, b, c\}$

ab	bac	baba	abbac	abbaab
ba	abcc	bacc	abccc	baabba
abc	baab	ababc	babac	babaabc

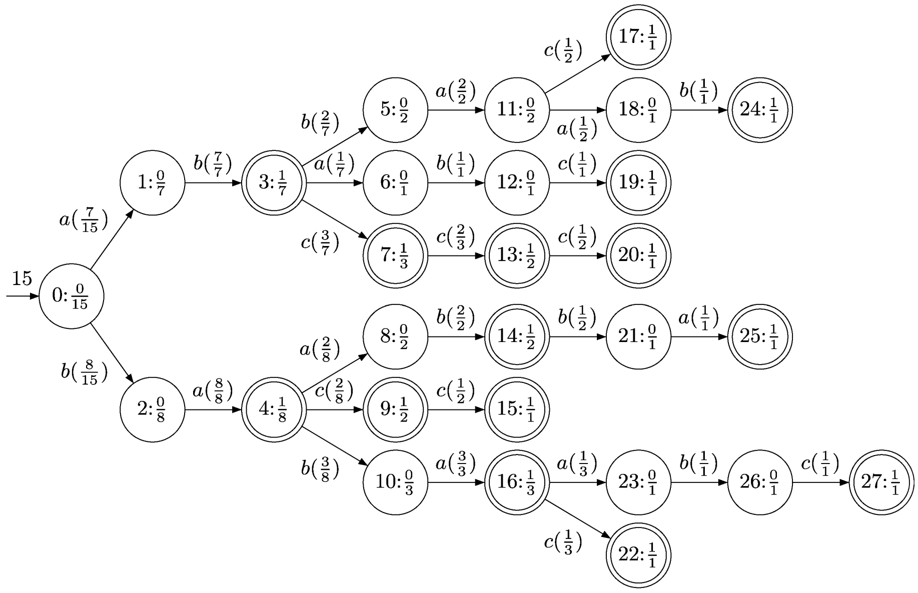


Fig. 6 PPTA corresponding to the sequences of Table 2

- $q: Q \times \Sigma \to Q$ is a *transition function*;
- q_0 is the initial state;
- $\pi: Q \times \Sigma \to [0, 1]$ is a probability function on the transitions;
- $\pi_F: Q \to [0, 1]$ is a probability function assigning to each state a probability to be final.

Suppose we have the set of 15 sequences presented in Table 2. Let us explain how we can build a PDFA which generalizes these sequences.

Among the inference algorithms able to learn PDFA, ALERGIA (Carrasco and Oncina 1994) is probably the most famous one. Using the set of sequences of Table 2, the principle of this algorithm is the following: ALERGIA first builds a probabilistic prefix tree acceptor (PPTA) which exactly models the input sequences (see Fig. 6). For each sequence, there is actually a unique path from the initial state 0, to a final state, symbolized by a double circle. To illustrate Definition 1, Figure 6 shows a PDFA where $Q = \{0, 1, 2, 3, \dots, 27\}$, $\Sigma = \{a, b, c\}$, $q_0 = 0$, and for instance $q(0, a) = 1$, $\pi(0, a) = \frac{7}{15}$ and $\pi_F(4) = \frac{1}{8}$.

Each transition is characterized by a symbol and a probability, representing respectively the emitted symbol and the proportion of sequences coming from the previous state and following that transition. For instance, the transition from state 0 to 1 has a probability of $\frac{7}{15}$ because among fifteen sequences entering in state 0, seven go to state 1. In this example, we represent on each node, not only its number but also its probability to be final. For example, state 3 has a probability of $\frac{1}{7}$ to be final, that means among the seven sequences entering, one ends in this state.

As we mentioned before, this automaton only models the set of sequences of Table 2. Let us recall that for overcoming the drawback of sets of sequences whose size does not exceed the lower bound N_{low}, we suggest generalizing the initial sequences. To do so, or in other words to avoid an overfitting phenomenon, ALERGIA works by state merging. It means that states are chosen in a lexicographical order and if they are sufficiently "similar", according to a compatibility function, they are merged. This function tests if the frequencies of each symbol outgoing from the two considered states are not statistically different. Based on Hoeffding's bound (Hoeffding 1963), this test decides that two states q_1 and q_2 can be merged if and only if:

$$\forall z \in \Sigma \cup \{\#\} \quad |\pi(q_1, z) - \pi(q_2, z)| < \sqrt{\frac{1}{2}\ln\left(\frac{2}{\alpha}\right)} \times \left(\frac{1}{\sqrt{n(q_1)}} + \frac{1}{\sqrt{n(q_2)}}\right) \tag{12}$$

where α is a generalization parameter, $n(q_1)$ and $n(q_2)$ are respectively the number of sequences entering in q_1 and q_2, and # is the termination symbol of a sequence. Hoeffding's bound is also checked for the successors states of q_1 and q_2. In order to obtain a deterministic automaton, other states have to be merged recursively. Let us explain this merging process from the PPTA of Fig. 6. Considering the merge of states 0 and 3, suppose we carry out Hoeffding's test with $\alpha = 0.8$. First, the Hoeffding's bound is calculated:

$$\sqrt{\frac{1}{2}\ln\left(\frac{2}{\alpha}\right)} \times \left(\frac{1}{\sqrt{n(0)}} + \frac{1}{\sqrt{n(3)}}\right) = \sqrt{\frac{1}{2}\ln\left(\frac{2}{0.8}\right)} \times \left(\frac{1}{\sqrt{15}} + \frac{1}{\sqrt{7}}\right) = 0.43.$$

Then, for each letter $z \in \Sigma \cup \{\#\}$, we calculate the difference of frequencies $|\pi(0, z) - \pi(3, z)|$. For $z = a$, we get $|\pi(0, a) - \pi(3, a)| = |\frac{7}{15} - \frac{1}{7}| = 0.32 < 0.43$. For $z = b$: $|\pi(0, b) - \pi(3, b)| = |\frac{8}{15} - \frac{2}{7}| = 0.25 < 0.43$. For $z = c$: $|\pi(0, c) - \pi(3, c)| = |\frac{0}{15} - \frac{3}{7}| = 0.42 < 0.43$. For $z = \#$: $|\pi(0, \#) - \pi(3, \#)| = |\frac{0}{15} - \frac{1}{7}| = 0.14 < 0.43$.

So states 0 and 3 can be merged, according to the compatibility constraint (see (12)). Figure 7 shows the resulting automaton (we will explain later the calculation of the transition probabilities, that is why we do not mention them in that figure).

We can note that this automaton is not deterministic because there are two transitions from state 0 labeled with the letter b ($q(0, b) = 5$ and $q(0, b) = 2$) and two transitions from state 0 labeled with the letter a ($q(0, a) = 1$ and $q(0, a) = 6$). So we need to merge the arrival states 2 with 5 and 1 with 6. If the resulting automaton is not yet deterministic, the merging process recursively continues to obtain the automaton of Fig. 8 (where the states have been renamed).

Let us now explain how to compute the new transition probabilities of the resulting automaton. Let $A' = \langle Q', \Sigma, q', q'_0, \pi', \pi'_F\rangle$ be the PDFA obtained after a given state merging from the PDFA $A = \langle Q, \Sigma, q, q_0, \pi, \pi_F\rangle$. Suppose that state $S' \in Q'$ is the result of the merge of the set of states $\{S_j, S_j \in Q\}$, we can calculate the probabilities as follows:

$$\pi(S', z) = \frac{\sum_j n(S_j, z)}{\sum_j \sum_{z' \in \Sigma\cup\{\#\}} n(S_j, z')}, \quad \forall z \in \Sigma \cup \{\#\},\ \forall S' = \{S_j, S_j \in Q\} \tag{13}$$

where $n(S_j, z)$ is the number of sequences outgoing from state S_j and following the transition labeled by the symbol z. Note that

$$\pi_F(S') = \pi(S', \#), \quad \forall S' = \{S_j, S_j \in Q\}. \tag{14}$$

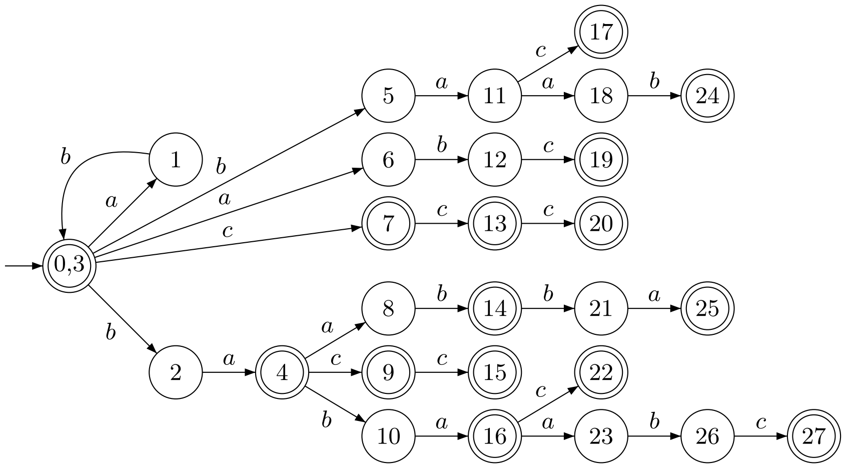


Fig. 7 Merge of state 0 and state 3

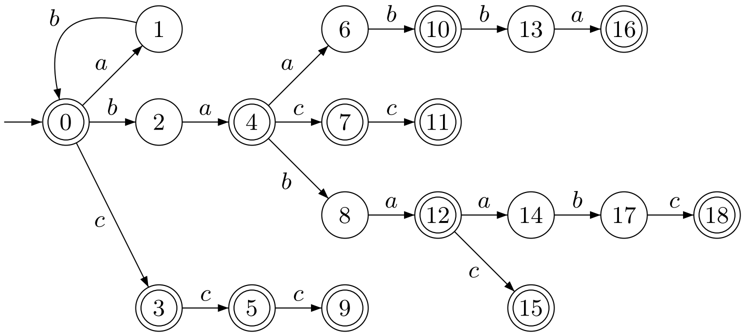


Fig. 8 Result of the merging process of states 0 and 3 after several recursive merges and the renaming of states

For example, state 0 of Fig. 8 is the result of the merge of states 0 and 3 of the automaton of Fig. 6. In fact, during the merging process, the state 12 has also been merged with state 0 in order to obtain a deterministic automaton, so we get:

$$\begin{aligned}\pi(0,a) &= \frac{n(0,a)+n(3,a)+n(12,a)}{\sum_{z'\in\Sigma\cup\{\#\}} n(0,z') + \sum_{z'\in\Sigma\cup\{\#\}} n(3,z') + \sum_{z'\in\Sigma\cup\{\#\}} n(12,z')}\\ &= \frac{7+1+0}{15+7+1} = \frac{8}{23},\\ \pi_F(0) &= \pi(0,\#)\\ &= \frac{n(0,\#)+n(3,\#)+n(12,\#)}{\sum_{z'\in\Sigma\cup\{\#\}} n(0,z') + \sum_{z'\in\Sigma\cup\{\#\}} n(3,z') + \sum_{z'\in\Sigma\cup\{\#\}} n(12,z')}\\ &= \frac{0+1+0}{15+7+1} = \frac{1}{23}.\end{aligned}$$

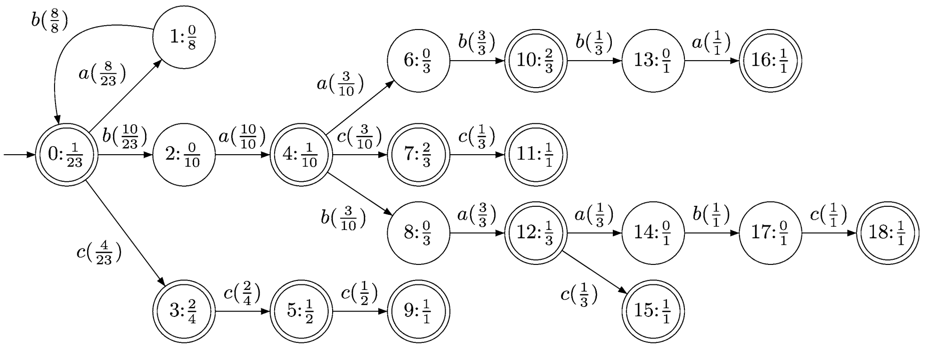


Fig. 9 New transition probabilities after the merge of states 0 and 3

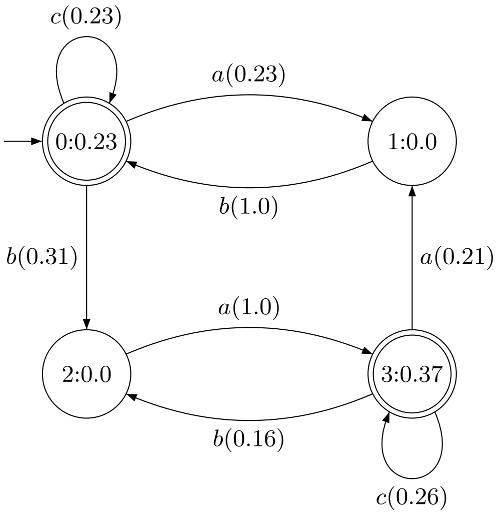


Fig. 10 Final PDFA inferred by ALERGIA from sequences of Table 2

Following this principle, we calculate all the probabilities of the automaton of Fig. 8 to obtain the PDFA of Fig. 9.

ALERGIA stops when no more merge is possible according to the compatibility function. Figure 10 shows the resulting final PDFA. We can see that all sequences of Table 2 are modeled by this PDFA, *i.e.* their parsing ends in a final state. Moreover, thanks to the merging process we can note that this PDFA generalizes the data. In other words, it means that it can represent other sequences that were not in the learning set (for instance, the sequence "ccab" is also modeled). A PDFA also provides a compact representation of the data that can be efficiently used in a sequence mining approach.

3.3 Estimation of pattern probabilities from a PDFA

If we aim to use a probabilistic automaton (learned from a set *LS* of sequences) to achieve a relevant sequence mining task, it must enable us to correctly estimate the true probability p of any pattern according to the distribution from which *LS* has been drawn. Hingston (2002) proposed a method to compute such estimates from a PDFA. Since we are going to use and then extend this approach, let us present the necessary background in the following section. We will then present the conditions to learn a good PDFA able to provide good estimates.

3.3.1 *Probability estimate of a symbol*

Let $A = \langle Q, \Sigma, q, q_0, \pi, \pi_F \rangle$ be a PDFA that has been learned (using for instance ALERGIA) from a learning set *LS* of sequences. To estimate from A the real probability $p(x)$ of sequences that contain a letter x, let $P(S, x)$ be the probability that a path in A starting from state S contains an x. This is ensured either if a path begins with an x (of probability $\pi(S, x)$), or with some other symbol $z \in \Sigma$ and is followed by a path starting at the next state (given by $q(S, z)$) and containing an x. This can be written with the recursive formula:

$$P(S,x) = \pi(S,x) + \sum_{z \neq x \in \Sigma} (\pi(S,z) \times P(q(S,z),x)), \tag{15}$$

that we can rewrite as follows:

$$P(S,x) = \pi(S,x) + \sum_{T \in Q} \left(\sum_{z \neq x, q(S,z)=T} \pi(S,z) \right) \times P(T,x). \tag{16}$$

If $S = q_0$, $P(S, x)$ represents the estimate of $p(x)$. Computing $P(S, x)$ requires handling a system of linear equations that can be efficiently solved with matrix products. Let $\rho(x)$ be the matrix of components

$$\rho_{S,T}(x) = \sum_{z \neq x, q(S,z)=T} \pi(S,z).$$

$\rho_{S,T}(x)$ simply describes the probability to use a transition different from x between states S and T. Let $P(x)$ (resp. $\pi(x)$) be the vector of values of $P(S, x)$ (resp. $\pi(S, x)$), (16) becomes:

$$P(x) = \pi(x) + \rho(x) \times P(x) = (I - \rho(x))^{-1} \times \pi(x), \tag{17}$$

where I is the identity matrix. Let us take an example with PDFA of Fig. 10 and estimate with $P(q_0, c)$ the unknown probability $p(c)$ of sequences that contain the letter c. Vector $\pi(c)$ has the components $\pi(0, c) = 0.23$, $\pi(1, c) = 0.0$, $\pi(2, c) = 0.0$, $\pi(3, c) = 0.26$, so we have

$$\pi(c) = \begin{pmatrix} 0.23 \\ 0 \\ 0 \\ 0.26 \end{pmatrix}.$$

For matrices $\rho(c)$ and $(I - \rho(c))^{-1}$, we get

$$\rho(c) = \begin{pmatrix} 0 & 0.23 & 0.31 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.21 & 0.16 & 0 \end{pmatrix}$$

and

$$(I-\rho(c))^{-1}=\begin{pmatrix}1.44 & 0.44 & 0.53 & 0.53\\ 1.44 & 1.44 & 0.53 & 0.53\\ 0.36 & 0.36 & 1.32 & 1.32\\ 0.36 & 0.36 & 0.32 & 1.32\end{pmatrix}.$$

So we get,

$$P(c)=\begin{pmatrix}1.44 & 0.44 & 0.53 & 0.53\\ 1.44 & 1.44 & 0.53 & 0.53\\ 0.36 & 0.36 & 1.32 & 1.32\\ 0.36 & 0.36 & 0.32 & 1.32\end{pmatrix}\times\begin{pmatrix}0.23\\ 0\\ 0\\ 0.26\end{pmatrix}=\begin{pmatrix}0.469\\ 0.469\\ 0.426\\ 0.426\end{pmatrix}.$$

We deduce that $P(0,c)=0.469$.

3.3.2 *Probability estimate of a pattern*

Based on the same principle, one can estimate the probability $p(w)$ of a pattern $w=\langle x_1\cdots x_l\rangle$ made up of l symbols *potentially nonconsecutive*. Let $F(S,T,x_1)$ be the probability that a random path starting at state S and ending at state T contains exactly one symbol x_1 at the last position. Hingston uses similar reasoning as in $P(x)$ to show that:

$$F(S,T,x_1)=\begin{cases}\sum_{x_j\neq x_1}(\pi(S,x_j)\times F(q(S,x_j),T,x_1))+\pi(S,x_1)\\ \quad \text{if } q(S,x_1)=T,\\ 0 \quad \text{otherwise.}\end{cases} \tag{18}$$

One can rearrange (18) using a matrix $\gamma(x_1)$ of values

$$\gamma(S,T,x_1)=\begin{cases}\pi(S,x_1) & \text{if } q(S,x_1)=T,\\ 0 & \text{otherwise.}\end{cases}$$

Writing $F(x_1)$ for the matrix of values $F(S,T,x_1)$, (18) becomes: $F(x_1)=\gamma(x_1)+\rho(x_1)\times F(x_1)$ and as before, we deduce

$$F(x_1)=(I-\rho(x_1))^{-1}\times\gamma(x_1).$$

Let us explain how to compute $P(q_0,\langle x_1\cdots x_l\rangle)$ the estimate of the probability $p(\langle x_1\cdots x_l\rangle)$. First, focus on the case $l=2$, *i.e.* $P(S,\langle x_1x_2\rangle)$. Note that a sequence containing an x_1 followed later by an x_2 can be divided into one part containing the first x_1 in the sequence, and the following part, which contains an x_2. We can deduce that:

$$P(S,\langle x_1x_2\rangle)=\sum_T F(S,T,x_1)\times P(T,x_2). \tag{19}$$

Using a matrix form, we get $P(\langle x_1x_2\rangle)=F(x_1)\times P(x_2)$. Generalizing this principle to l symbols, we get

$$P(\langle x_1\cdots x_l\rangle)=F(x_1)\times\cdots\times F(x_{l-1})\times P(x_l). \tag{20}$$

For example, with the PDFA of Fig. 10, we can estimate the probability $p(\langle cc\rangle)$ of sequences containing the pattern $\langle cc\rangle$. To do so, we have to compute $P(0, \langle cc\rangle)$. We need $\gamma(c)$:

$$\gamma(c)=\begin{pmatrix}0.23 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0. & 0 & 0 & 0.26\end{pmatrix}$$

to calculate $F(c)$:

$$F(c)=\begin{pmatrix}1.44 & 0.44 & 0.53 & 0.53\\ 1.44 & 1.44 & 0.53 & 0.53\\ 0.36 & 0.36 & 1.32 & 1.32\\ 0.36 & 0.36 & 0.32 & 1.32\end{pmatrix}\times\begin{pmatrix}0.23 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0.26\end{pmatrix}=\begin{pmatrix}0.33 & 0 & 0 & 0.14\\ 0.33 & 0 & 0 & 0.14\\ 0.08 & 0 & 0 & 0.34\\ 0.08 & 0 & 0 & 0.34\end{pmatrix}.$$

So

$$P(\langle cc\rangle)=\begin{pmatrix}0.33 & 0 & 0 & 0.14\\ 0.33 & 0 & 0 & 0.14\\ 0.08 & 0 & 0 & 0.34\\ 0.08 & 0 & 0 & 0.34\end{pmatrix}\times\begin{pmatrix}0.469\\ 0.469\\ 0.426\\ 0.426\end{pmatrix}=\begin{pmatrix}0.21\\ 0.21\\ 0.18\\ 0.18\end{pmatrix}.$$

We deduce that $P(0, \langle cc\rangle) = 0.21$.

Although one can justify the use of a PDFA by its ability to generalize the set *LS* of sequences, an important question remains: Is the learned PDFA a good generative model? In other words, since we use ALERGIA, what are the conditions to iteratively achieve good state merging? Hingston (Hingston 2002) does not deal with this problem. We provide a formal answer to this question in the following section.

4 How to learn a good PDFA for sequence mining?

4.1 Theoretical results

In the grammatical inference literature, many papers deal with the learnability of regular languages in the form of PDFA (see de la Higuera 2005 for a survey). All the formal results, such as the identification in the limit (Gold 1978) or the PAC learnability (Valiant 1984), give a theoretical framework to guarantee the learnability of PDFA from a set of sequences. In both cases, the goal is to provide the conditions on the number of sequences to learn a *target concept*. Rather than imposing such a strong constraint, we are interested in the definition of a bound on the number of symbols ensuring that the risks of wrongly accepting and rejecting a merge with ALERGIA are respectively bounded by two parameters α and β.

During the merging process of ALERGIA, the Hoeffding's bound is applied to test the compatibility of two states (see Sect. 3.2). The generalization parameter α, used in the test, is nothing else but a Type I error of a test of estimation error. We can note that the Type II error β is not used in that test. In other words, the only risk that is controlled by ALERGIA is the risk to wrongly reject a good merge. In (Carrasco and Oncina 1994), a theoretical result shows that at the limit, both α and β risks decrease with the increase in the number of symbols. But, nothing is said about the minimal number of symbols, considered by the merge, that guarantees some given risks α and β. We provide such a bound in this section.

Let us recall that two states q_1 and q_2 are merged in ALERGIA *iff*:

$$\forall z \in \Sigma \cup \{\#\}$$
$$|\pi(q_1, z) - \pi(q_2, z)| < \sqrt{\frac{1}{2}\ln\left(\frac{2}{\alpha}\right)} \times \left(\frac{1}{\sqrt{n(q_1)}} + \frac{1}{\sqrt{n(q_2)}}\right).$$

$\pi(q_1, z)$ and $\pi(q_2, z)$ are estimates of true probabilities $p(q_1, z)$ and $p(q_2, z)$. To take into account not only α but also β, we suggest the use of a test of proportions instead of the Hoeffding's bound to assess the compatibility of two states. Since a good merge occurs when $\forall z p(q_1, z) = p(q_2, z)$, we test the null hypothesis H_0: $p(q_1, z) - p(q_2, z) = 0$ against the alternative one H_a: $p(q_i, z) - p(q_j, z) > 0$. Note that the direction of the test will depend on the observed data. In other words, to have a one-tailed test, q_i will be the state (q_1 or q_2) from which the highest proportion of z is observed. More formally, $i = \arg\max_{k\in\{1,2\}} \pi(q_k, z)$. If $n(q_1) > 15$ and $n(q_2) > 15$ it is common knowledge that $\pi(q_i, z) - \pi(q_j, z)$ follows a normal distribution:[1]

$$\pi(q_i, z) - \pi(q_j, z)$$
$$\approx \mathcal{N}\left(p(q_i, z) - p(q_j, z), \sqrt{\frac{p(q_i, z) \times \bar{p}(q_i, z)}{n(q_i)} + \frac{p(q_j, z) \times \bar{p}(q_j, z)}{n(q_j)}}\right),$$

where $\bar{p}(q_k, z) = 1 - p(q_k, z)$. Under H_0, $p(q_1, z) = p(q_2, z) = p(q, z)$. Since $p(q, z)$ is unknown, we can estimate it by $\pi(q, z)$ such that:

$$\pi(q, z) = \frac{\pi(q_1, z) \times n(q_1) + \pi(q_2, z) \times n(q_2)}{n(q_1) + n(q_2)}. \tag{21}$$

Let α be the Type I error, *i.e.* the risk to refuse a good merge. This will be done if the difference $\pi(q_i, z) - \pi(q_j, z)$ exceeds a rejection bound k, over which it only remains $\alpha\%$ of the density of the normal distribution that satisfies H_0.

$$\alpha = P(H_a|H_0) = P\left(\pi(q_i, z) - \pi(q_j, z) > k|H_0\right). \tag{22}$$

Subtracting the mean $p(q_i, z) - p(q_j, z)$ and dividing by the standard deviation $\sqrt{\frac{p(q_i,z)\times\bar{p}(q_i,z)}{n(q_i)} + \frac{p(q_j,z)\times\bar{p}(q_j,z)}{n(q_j)}}$, we obtain a centered and reduced variable Z that follows the normal distribution $\mathcal{N}(0, 1)$. Using the estimate of (21), we get

$$\alpha = P\left(Z > \frac{k}{\sqrt{\pi(q, z) \times \bar{\pi}(q, z)} \times \sqrt{\frac{1}{n(q_i)} + \frac{1}{n(q_j)}}}\right), \tag{23}$$

where $\bar{\pi}(q, z) = 1 - \pi(q, z)$. Let β be the Type II error, *i.e.* the risk to accept a wrong merge. Under H_a, $p(q_i, z) - p(q_j, z) = p_a > 0$. We get:

$$\beta = P(H_0|H_a) = P\left(Z < \frac{k - p_a}{\sqrt{\frac{\pi(q_i,z)\times\bar{\pi}(q_i,z)}{n(q_i)} + \frac{\pi(q_j,z)\times\bar{\pi}(q_j,z)}{n(q_j)}}}\right). \tag{24}$$

[1] For small numbers of sequences, we can use a Fisher exact test (Fisher 1922). See Sect. 5.2.2 for more details.

From (23) and (24), it implies that:

$$k = z_\alpha \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{1}{n(q_i)} + \frac{1}{n(q_j)}} \tag{25}$$

and

$$k = p_a - z_\beta \times \sqrt{\frac{\pi(q_i,z) \times \bar{\pi}(q_i,z)}{n(q_i)} + \frac{\pi(q_j,z) \times \bar{\pi}(q_j,z)}{n(q_j)}}, \tag{26}$$

where z_α (resp. z_β) corresponds to the $(1-\alpha)$ (resp. $(1-\beta)$) percentile of the normal distribution.

Theorem 2 *To ensure that the proportions of rejected good merges and accepted wrong ones do not exceed respectively fixed risks α and β, the minimal number of symbols n_{low} on which the merging process must be run is equal to*

$$n_{low} = \frac{1+\gamma}{p_a^2} \times \Bigg(z_\alpha \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{\gamma+1}{\gamma}} + z_\beta \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}} \Bigg)^2,$$

where γ represents the observed ratio $\frac{n(q_j)}{n(q_i)} > 0$.

Proof The proof is straightforward. We replace $n(q_j)$ by $\gamma \times n(q_i)$ in (25) and (26):

$$\begin{aligned} k &= \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \frac{1}{\sqrt{n(q_i)}} \times \sqrt{\frac{\gamma+1}{\gamma}} \times z_\alpha \\ &= p_a - z_\beta \times \frac{1}{\sqrt{n(q_i)}} \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}} \end{aligned}$$

$\Leftrightarrow$

$$p_a = \frac{1}{\sqrt{n(q_i)}} \times \Bigg(z_\alpha \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{\gamma+1}{\gamma}} + z_\beta \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}} \Bigg).$$

Extracting $n(q_i)$, we obtain the lower bound $n_{low}(q_i)$ on the number of symbols entering in the state q_i.

$$n_{low}(q_i) = \frac{1}{p_a^2} \times \Bigg(z_\alpha \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{\gamma+1}{\gamma}} + z_\beta \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}} \Bigg)^2.$$

The minimal number of symbols that must be concerned by the state merging is then $n_{low} = n_{low}(q_i) + n_{low}(q_j) = (1+\gamma) \times n_{low}(q_i)$ that gives the lower bound. □

4.2 Example

Let us provide an example of the calculation of this bound. If we consider the part of a PDFA of Fig. 11, let us assume we test the possible merge between states 1 and 2. Let us calculate the number n_{low} of *local symbols* necessary to achieve a good merge.

Note that for each symbol $a, b, \#$ concerned by this merge (let us recall that # is a termination symbol of a string), we have to calculate the bound. We have $\pi(1,a) = \frac{70}{140}$ and $\pi(2,a) = \frac{63}{129}$. Let us fix $p_a = 0.15$ and $\alpha = \beta = 5\%$, so $z_\alpha = z_\beta = 1.64$ (looking at the table of the normal distribution). The calculation of (21) for the letter a gives

$$\begin{aligned}\pi(q,a) &= \frac{\pi(1,a) \times n(1) + \pi(2,a) \times n(2)}{n(1) + n(2)} \\ &= \frac{\frac{70}{140} \times 140 + \frac{63}{129} \times 129}{140 + 129} = \frac{133}{269}.\end{aligned}$$

As $\gamma = \frac{129}{140} = 0.92$, we get:

$$\begin{aligned}n_{low} &= \frac{1+\gamma}{p_a^2} \times \Bigg(z_\alpha \times \sqrt{\pi(q,a) \times \bar{\pi}(q,a)} \times \sqrt{\frac{\gamma+1}{\gamma}} \\ &\quad + z_\beta \times \sqrt{\frac{(\gamma \times \pi(1,a) \times \bar{\pi}(1,a)) + \pi(2,a) \times \bar{\pi}(2,a)}{\gamma}} \Bigg)^2 \\ &= \frac{1+0.92}{0.15^2} \times \Bigg(1.64 \times \sqrt{\frac{133}{269} \times \frac{136}{269}} \times \sqrt{\frac{0.92+1}{0.92}} \\ &\quad + 1.64 \times \sqrt{\frac{(0.92 \times \frac{70}{140} \times \frac{70}{140}) + \frac{63}{129} \times \frac{66}{129}}{0.92}} \Bigg)^2 \\ &= 207.\end{aligned}$$

For b and # we obtain respectively the lower bounds 207 and 79. So we can decide that states 1 and 2 deserve to be merged because $n(1) + n(2) = 269$ that satisfies the three lower bounds 207, 207 and 79.

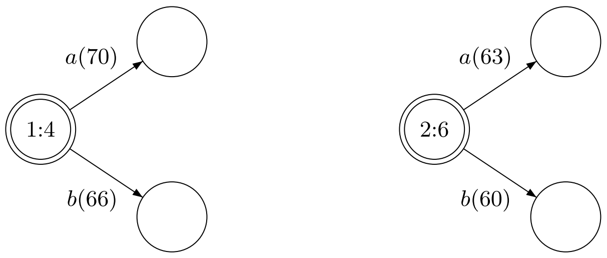


Fig. 11 States 1 and 2 within a PDFA are candidates to be merged. There are 4 sequences that end in state 1 and 6 sequences in state 2

4.3 Experimental validation

To assess the efficiency of our lower bound n_{low} we carried out the following series of experiments: We used the Reber grammar (Reber 1967) to sample sets of sequences LS_i of increasing sizes. For each of them, we learned two PDFA:

- A first one inferred by ALERGIA and its standard Hoeffding's bound.
- A second one learned from a modified version of ALERGIA, combining the Hoeffding's bound and our lower bound n_{low} (using $\alpha = \beta = 5\%$). This means that the two bounds must be satisfied to accept a state merging.

From these automata, we respectively computed $P_H(q_0, w)$ and $P_{H+n_{low}}(q_0, w)$ for all the patterns w of 1, 2 and 3 symbols, *i.e.* $\forall w \in (\Sigma \cup \{\lambda\})^3$ (where λ is the empty symbol), and compared them with the true probability $p(w)$ calculated from the target distribution (*i.e.* the Reber grammar). On the other hand, we computed the observed proportions $\hat{p}(w)$ of each pattern directly from the sets LS_i, and we also compared it with $p(w)$. Let us recall that $\hat{p}(w)$ is the information used by the classical sequence mining algorithms. Figure 12 shows the normalized average difference between the estimated and the true probabilities. Two main remarks can be made:

- First, the two curves concerning the use of a PDFA converge toward 0; this means that with an increasing number of sequences LS_i, the number of symbols concerned by the state merging obviously also increases, leading to better decisions during the merging process. However, Hoeffding's bound alone is not sufficient, and a joint use with our theoretical bound n_{low} allows to avoid wrong decisions, particularly when the number of symbols is not large. In these cases, the test of our bound often leads to the rejection of wrong state merging resulting in better estimates.
- Second, the curve drawn from the sequences of LS_i also decreases. However, the estimates computed from the PDFA with our lower bound are always better than those calculated from the sequences themselves. It confirms that PDFA-based sequence mining

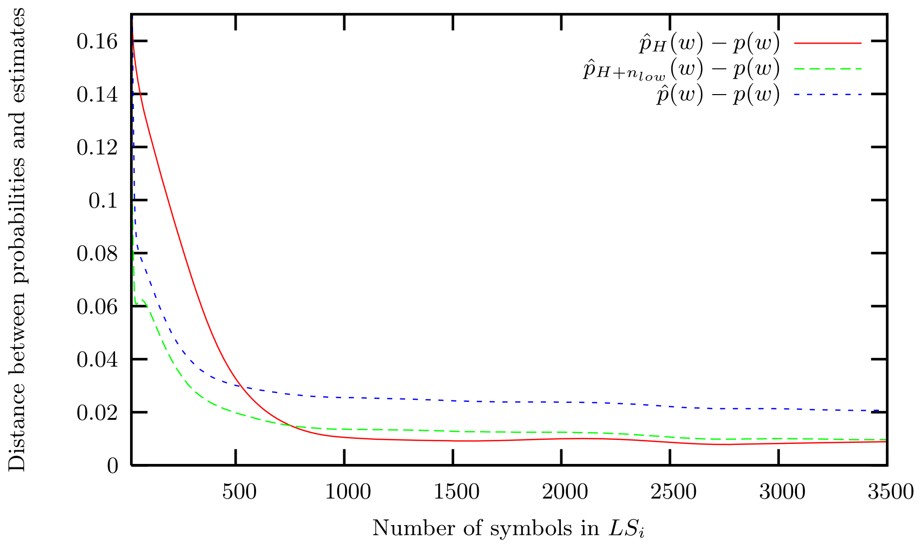


Fig. 12 Average difference between $P(q_0, w)$ and $p(w)$ on the Reber grammar

approach can constitute a good alternative to standard algorithms, whatever the number of sequences.

5 Constrained sequence mining

The use of constraints is one of the current trends in sequence mining. Length and width restrictions, minimum or maximum gap between elements, time window of occurrence, or regular expressions (Zaki 2000; Garofalakis et al. 2002; Pei et al. 2002) are used to reduce the (potentially huge) number of extracted frequent patterns. We present two types of PDFA-based constraints. The first one (Sect. 5.1) is directly devoted to reduce the number of extracted patterns. Only patterns that occur in the PDFA after prefixes of a given length are extracted. The second constraint we present only extracts the frequent patterns that are statistically significant. In other words, we provide in this section the last statistical tool that helps the user to extract true knowledge. Actually, let us suppose that neither the lower bound N_{low} (Sect. 2), nor the lower bound n_{low} (Sect. 4) has been satisfied. Are we still able to extract significant knowledge from the automaton? We provide a solution to this problem in Sect. 5.2.

5.1 Prefix length constraint

We introduce a constraint which enables us to discover patterns satisfying a given prefix-length. This can be mainly interesting in domains where the location of the patterns in the sequence expresses their meaning. It is the case, for instance, in bioinformatics, for the transcription factor binding sites. Let us now explain the computation of the probability of such constrained patterns. Let $P(S,x,\theta)$ represent the proportion of sequences containing an x (maybe not the first one) at a distance θ from state S. Let $P(S,x,\theta)$ be a component of the vector $P(x,\theta)$. For example, from Fig. 10, if we are looking for the proportion $P(0,a,2)$ of sequences containing an a at a distance two, we can establish that:

$$\begin{aligned}P(0,a,2) &= \pi(0,a)\times\pi(1,b)\times\pi(0,a)+\pi(0,c)\times\pi(0,c)\times\pi(0,a)\\ &\quad +\pi(0,c)\times\pi(0,b)\times\pi(2,a)+\pi(0,b)\times\pi(2,a)\times\pi(3,a)\\ &= 0.23\times 1.0\times 0.23+0.23\times 0.23\times 0.23\\ &\quad +0.23\times 0.31\times 1.0+0.31\times 1.0\times 0.21\\ &= \sum_{z\in\Sigma}\pi(0,z)\times P(q(0,z),a,1)=0.201.\end{aligned}$$

By observing the original sequences of Table 2, we can note that the proportion of sequences that contain a in the third position is $\frac{3}{15}=0.2$.

Generalizing, we get:

$$\begin{aligned}P(S,x,\theta) &= \sum_{z\in\Sigma}\pi(S,z)\times P(q(S,z),x,\theta-1) && (27)\\ &= \sum_{T\in Q}\Bigg(\sum_{z,q(S,z)=T}\pi(S,z)\Bigg)\times P(T,x,\theta-1). && (28)\end{aligned}$$

Let $\tau_{S,T} = \sum_{z,q(S,z)=T} \pi(S,z)$ be the probability to use one transition between states S and T. Using the values $\tau_{S,T}$, we get:

$$P(S,x,\theta) = \sum_{T \in Q} \tau_{S,T} \times P(T,x,\theta-1). \tag{29}$$

Let $P(x,\theta)$ be the vector of values of $P(S,x,\theta)$, (29) becomes: $P(x,\theta) = \tau \times P(x,\theta-1)$. This is a geometric series of common ratio τ (the matrix of values $\tau_{S,T}$) and first term $\pi(S,x)$. Plugging $\pi(x)$ in $P(x,\theta)$, we get:

$$P(x,\theta) = \tau^{\theta} \times \pi(x). \tag{30}$$

It is possible to generalize $P(S,x,\theta)$ to $P(S,w,\theta)$, the probability to encounter any pattern w at a distance θ from state S. Let us focus on the case $w = \langle x_1 x_2 \rangle$ *i.e.* $P(S,\langle x_1 x_2 \rangle,\theta)$. We can represent the position θ of the pattern w by the position of its first letter. Note that a sequence containing an x_1 at the position θ followed later by an x_2 can be divided into one part containing the x_1 at position θ in the sequence, and the following part which contains an x_2. Let $F(S,T,x_1,\theta)$ be the probability that a random path starting at state S and ending at state T contains the symbol x_1 which is the θth symbol on the path.

As done in Sect. 3.3 we can show that:

$$F(S,T,x_1,\theta) = \sum_{z \in \Sigma} \pi(S,z) \times F(q(S,z),T,x_1,\theta-1) \tag{31}$$

$$= \sum_{R \in Q} \left(\sum_{z,q(S,z)=R} \pi(S,z) \right) \times F(R,T,x_1,\theta-1). \tag{32}$$

Using the values $\tau_{S,T}$ previously defined, we get:

$$F(S,T,x_1,\theta) = \sum_{R \in Q} \tau_{S,R} \times F(R,T,x_1,\theta-1). \tag{33}$$

Writing $F(x,\theta)$ the vector of values of $F(S,T,x,\theta)$, (33) becomes: $F(x,\theta) = \tau \times F(x,\theta-1)$. This is a geometric series of common ratio τ and first term $\gamma(x)$ (see (18)). We get

$$F(x,\theta) = \tau^{\theta} \times \gamma(x). \tag{34}$$

We can deduce that

$$P(S,\langle x_1 x_2 \rangle,\theta) = F(x_1,\theta) \times P(x_2). \tag{35}$$

Generalizing this principle to l symbols, we get

$$P(\langle x_1 \cdots x_l \rangle,\theta) = F(x_1,\theta) \times F(x_2) \times \cdots \times F(x_{l-1}) \times P(x_l). \tag{36}$$

We could think that this prefix length constraint could sometimes be too strong. Actually, we mentioned that it could be used in bioinformatics. It is known that, in DNA sequences, mutations can occur, so the search for a pattern at a fixed position could be irrelevant. In order to relax this constraint, we introduce the stack variable ϵ that enables the discovery of patterns at position $\theta \pm \epsilon$. Using the previous formulas, we can easily show that:

$$\begin{aligned}&P(\langle x_1 \cdots x_l\rangle, \theta \pm \epsilon)\\&\quad = P(\langle x_1 \cdots x_l\rangle, \theta) + \sum_{i=1}^{\epsilon} (P(\langle x_1 \cdots x_l\rangle, \theta + i) + P(\langle x_1 \cdots x_l\rangle, \theta - i)).\end{aligned} \tag{37}$$

5.2 Statistical significance of a pattern

Our aim is to use the values $P(q_0, \langle x_1 \cdots x_l\rangle)$ to assess the statistical significance of a frequent pattern. If the theoretical bounds presented in the previous sections are satisfied in a given application, the problem is definitely solved and the frequent patterns extracted from the PDFA are proved to be likely true knowledge. However, what happens if these bounds are not satisfied (for instance, n_{low} is not always verified during the state merging process)? We define the concept of *statistical significance* of a pattern. Since tuning the support threshold is a tricky task, we constrain, using two statistical tests, a sequence to be not only frequent but also statistically significant.

5.2.1 Proportion constraint

The first test verifies an *absolute* condition: a pattern $w = \langle x_1 \cdots x_l\rangle$ must cover a significant part of the probability density of all sequences. To fulfill this constraint, we apply a proportion test (called PROP_TEST) that aims to verify if $P(q_0, w)$ (the estimate of $p(w)$) is high enough. To do this, we test the null hypothesis H_0: $p(w) = 0$, against the alternative one H_a: $p(w) > 0$. If the number of sequences N is large enough (> 30), $P(q_0, \langle w\rangle)$ asymptotically follows the normal distribution. Let us determine the threshold k which defines the bound of rejection of H_0, and which corresponds to the $(1 - \alpha_1)$-percentile z_{α_1} of the distribution of $p(w)$ under H_0. We can show that

$$P\left(P(q_0, w) > k\right) = \alpha_1 \text{ iff } k = z_{\alpha_1}\sqrt{\frac{P(q_0, w)(1 - P(q_0, w))}{N}}.$$

We then get the decision rule: *if $P(q_0, w) > k$, the proportion constraint on w is satisfied.*

For example, using the sequences of Table 2, let us consider the pattern cc. In Sect. 3.3.2 we showed that $P(0, \langle cc\rangle) = 0.21$. Fixing $\alpha_1 = 5\%$, we get:

$$k = 1.64 * \sqrt{\frac{0.21 * (0.79)}{15}} = 0.172.$$

$P(q_0, \langle cc\rangle) = 0.21 > 0.172$ so the proportion constraint on cc is satisfied.

5.2.2 Dependence constraint

We also impose a relative condition; we test if there exists a statistical dependence between w and $w' = \langle x_1 \cdots x_{l-1}\rangle$. Roughly speaking, this dependence is satisfied if the majority of the sequences that contain w' also contain $w = w'.\langle x_l\rangle$, where "." is the concatenation function. This dependence can be assessed by analyzing the nature of x_l occurring after w'. We generate an output vector $\overrightarrow{V_{out}}$ of dimension $|\Sigma \cup \{\#\}|$. Each component $\overrightarrow{V_{out}}(i)$ is the expected number of sequences that have the symbol $z_i \in \Sigma \cup \{\#\}$ which follows the pattern w'. It means that $\overrightarrow{V_{out}}(i) = P(q_0, \langle x_1 \cdots x_{l-1} z_i\rangle) \times N$. We arrange the components of $\overrightarrow{V_{out}}$ in a

Table 3 2×2 contingency table. The objective of the Fisher exact test is to verify if $\overrightarrow{V_{in}}$ and $\overrightarrow{V_{out}}$ are realizations of the same random variable. We assume here that $\Sigma = \{x_l, x_{l'}\}$; but this can be extended to larger alphabets

			Total
$\overrightarrow{V_{in}}$	$N \times P(q_0, w')$	0	L_1
$\overrightarrow{V_{out}}$	$N \times P(q_0, w'.\langle x_l \rangle)$	$N \times P(q_0, w'.\langle x_{l'} \rangle)$	L_2
Total	C_1	C_2	$C_1 + C_2 + L_1 + L_2$

lexicographical order. Our goal is to test the dependence between $\overrightarrow{V_{out}}$ and an input vector $\overrightarrow{V_{in}}$ (ordered as $\overrightarrow{V_{out}}$) for which only one component is not null, corresponding to the expected number of sequences that contain the pattern w'. To do so, we consider the H_0 hypothesis that $\overrightarrow{V_{in}}$ and $\overrightarrow{V_{out}}$ are both a realization of the same multinomial random variable. Since some components of $\overrightarrow{V_{in}}$ are null, we cannot use the Pearson statistics (Pearson 1900). However, we can use a Fisher exact test (called FISHER_TEST). Let us explain the principle of this test.

Given the two vectors $\overrightarrow{V_{in}}$ and $\overrightarrow{V_{out}}$ and their components summarized in the contingency table of Fig. 3 (to simplify we consider a 2×2 contingency table but this can be extended to $2 \times |\Sigma|$ tables). The Fisher exact test enables us to compute the probability of all the contingency tables that have the same marginal counts (C_1, C_2, L_1, L_2) and that are at least unfavorable to H_0. Fisher (1922) showed that the probability of such contingency tables are given by a hyper-geometric distribution. If the sum of the probabilities of the previous matrices is smaller than a given risk α_2, H_0 is accepted, and then *the dependence constraint is verified.*

By combining the proportion and dependence constraints, we can now define a frequent and significant pattern.

Definition 2 A pattern $w = \langle x_1 \cdots x_{l-1} x_l \rangle$ is frequent and *significant iff* (i) $P(q_0, w)$ is higher than a support threshold p_0, (ii) the proportion constraint on $P(q_0, w)$ is satisfied with a risk α_1 and (iii) the dependence constraint between w and $w' = \langle x_1 \cdots x_{l-1} \rangle$ is satisfied with a risk α_2.

5.3 The ACSM algorithm

Combining our two constraints we presented so far, we suggest a new constrained sequence mining algorithm. The objective is to discover from a PDFA all the frequent and significant patterns, according to a support threshold p_0 and two statistical risks α_1 and α_2. The pseudo-code of our ACSM algorithm (Automata-based Constrained Sequence Mining) is presented in Algorithm 1. From lines 2 to 9, it initializes a set of significant frequent patterns composed of only one symbol. Since no pattern has been extracted yet, only the support test (line 4) and PROP_TEST (line 5) are run. The paths of the PDFA that do not satisfy these two tests will not be studied anymore, that allows us to prune the search space. The second part of ACSM tests if additional symbols can be added to generate larger frequent and significant patterns. Three conditions must be satisfied: the support test (lines 18), PROP_TEST (line 19) and FISHER_TEST (line 21). The boolean function MERGE(w, w') (line 16) returns *true* if the $n-1$ last symbols of w are identical to the $n-1$ first ones of w'. This ensures that all the subsequences of the resulting pattern $v = w.\langle x'_n \rangle$ (line 17) are already frequent and significant. In this version of the algorithm, we do not include the prefix length constraint for the sake of understandability. To do this we would simply have to introduce new parameters θ and ϵ, and replace the calculation of $P(q_0, v)$ by $P(q_0, v, \theta)$ in lines 5, 18 and 19.

```
Input: A PDFA $A = (Q, \Sigma, q, q_0, \pi, \pi_F)$, a support threshold $p_0$, two risks
       $\alpha_1$ and $\alpha_2$
Output: a set of significant frequent patterns G
1  begin
2    $G_1 \leftarrow \emptyset$ ;
3    foreach $l \in \Sigma$ do
4      if $P(q_0, l) > p_0$ then
5        if PROP_TEST $(P(q_0, l), \alpha_1)$ is satisfied then
6          $G_1 \leftarrow G_1 \cup \{l\}$ ;
7        end
8      end
9    end
10   $G \leftarrow G_1$ ;
11   $n \leftarrow 1$ ;
12   while $G_n \neq \emptyset$ do
13     $G_{n+1} \leftarrow \emptyset$ ;
14     foreach $w =< x_1 ... x_n > \in G_n$ do
15       foreach $w' =< x'_1 ... x'_n > \in G_n$ do
16         if MERGE $(w, w')$ then
17           $v \leftarrow w. < x'_n >$ ;
18           if $P(q_0, v) > p_0$ then
19             if PROP_TEST $(P(q_0, v), \alpha_1)$ then
20               Compute $\overrightarrow{V_{in}}$ and $\overrightarrow{V_{out}}$ /*see 5.2*/
21               if FISHER_TEST $(\overrightarrow{V_{in}}, \overrightarrow{V_{out}}, \alpha_2)$ then
22                 $G_{n+1} \leftarrow G_{n+1} \cup \{v\}$ ;
23               end
24             end
25           end
26         end
27       end
28     end
29     $G \leftarrow G \cup G_{n+1}$ ;
30     $n \leftarrow n + 1$ ;
31   end
32   return $G$ ;
33 end
```

Algorithm 1 Pseudo-code of ACSM

6 PDFA-based sequence mining: an original solution to privacy preservation

We present in this section our last contribution. Until now, we assumed that we had a set of sequences *LS* from which a PDFA is learned to model a regular language. The interest of such an approach is to represent more sequences than those of *LS* and to provide a compact representation of the data which can be understandable and from which a sequence mining task can be efficiently done. Another important advantage of such a PDFA-based representation is its potential exploitation when we do not want to (or we cannot) have access to the original sequences. Actually, as we mentioned in the introduction, privacy preservation is a new trend in sequence mining. It consists of the preservation of specific characteristics of the data. As stated in (Verykios et al. 2004), *the main objective in privacy preserving data mining is to develop algorithms for modifying the original data in some way, so that the private data and private knowledge remain private even after the mining process*. A huge number of papers have been published in that domain during the last five years, for example (Agrawal and Srikant 2000; Sweeney 2002; Evfimievski et al. 2004; Bayardo and Agrawal 2005), proposing different types of methods like data distribution or data modification in order to hide sensitive data.

We think that PDFA-based sequence mining, as presented in this paper, provides a very promising solution to privacy preservation for problems that may be stated as flow control problems. In order to take care of the sensitivity of a set of sequences, so as to preserve the privacy of the data, we can extract frequent sequences from a PDFA without having any information about the original data. The crucial question that remains is the following: How can we directly have a PDFA without learning it from sequences? Let us give two real world examples on which we can apply our PDFA-based approach for privacy preservation.

The first one concerns mining of Web sites. In that case, the states of the automaton are the pages of the site and transitions are the hyperlinks between pages. The weight assigned to each transition corresponds to the number of users who clicked on the corresponding hyperlink and the weight of each state is the number of times users left the web site from the corresponding page. We do not need anymore users' IP address and the Web pages they visited, but only counters on each transition and state. This allows us to find frequent paths within web site structures, without knowing the identity of users who take those paths.

The problem of discovering frequent routes in a town can also be directly achieved from an automaton. As we already mentioned, installing Web cams and tracing the routes of each driver is obviously a nonacceptable breach to their privacy, even using an anonymization process of the inputs. A PDFA-based approach overcomes this drawback. Actually, considering a map of a town, the non-initial and non-final states can model the crossroads. The initial and final states respectively represent the entry and exit gates in the map. The transitions model the streets. The weights are computed by using counters on the transitions and on the final states (the other states having a null counter), that is easier to obtain than sequences. For each car, we do not need anymore its license plate and the streets it traverses, and thus we do not have to put in cameras to follow each car. Despite the fact we do not have the individuals' routes, we are able with ACSM to extract significant frequent paths (potentially with gaps).

However, an important problem occurs when attempting to preserve individual privacy using a PDFA. So far, we provided theoretical conditions on the number of sequences or symbols to guarantee the relevance of an automaton. In this new context of privacy preservation, we do not have access anymore to the sequences themselves. Therefore, are we still able to assess the quality of the PDFA? We answer this question in the following section.

6.1 Assessment of the quality of a PDFA in the context of privacy preservation

In the best case, the transitions of the PDFA are described by the number of times they have been used by the sequences; this is the case in the two previously cited examples, *i.e.* web sites and road traffic. In such cases, knowing the number of times a transition has been used between two states and the number of times the outgoing transitions have been followed, we are able to compute the number N of sequences from whom the PDFA has been built. To assess the quality of the PDFA, we could suggest a comparison between N and the theoretical bound N_{low} presented in Sect. 2. However, N_{low} has been originally provided to assess the quality of a sequence mining algorithm. Even if we can think it also provides a good condition to assess the quality of the PDFA, this is not entirely satisfying.

In the worst case, no counts are available for the transitions in the PDFA but only probabilities; hence the number of sequences cannot be deduced and N_{low} cannot be computed to assess the quality of the PDFA. To overcome this drawback, we suggest in the following

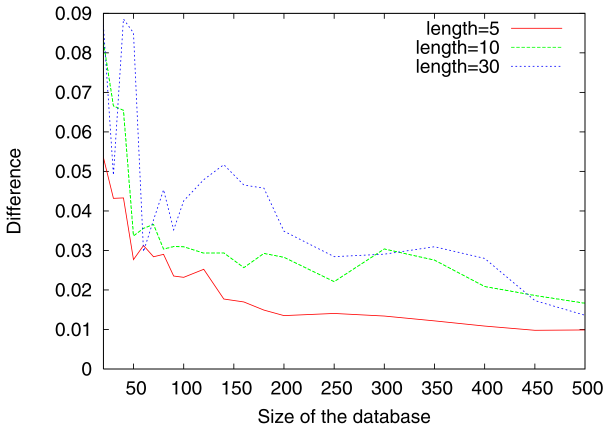


Fig. 13 Average difference between $P(q_0, w)$ and $p(w)$

the exploitation of another piece of information that can be extracted from the PDFA: the expected size of the sequences.

6.1.1 Effect of the length of the sequences on the quality of the PDFA

It is common knowledge in grammatical inference that the length of the sequences has a direct impact on the convergence of the inference algorithms. This can be explained by the fact that a PDFA has difficulties to model long time dependencies (Callut 2007) because it is based on the Markov property. Moreover, the larger the length of the sequences, the higher the overfitting risk of the inference grammatical algorithm. Without any information about the number of sequences, we show in this section that we are able to compute the expected length of the sequences modeled by a PDFA. We show that it provides good information about the quality of the model. In other words, the shorter the length, the more confident in the PDFA we are. Let us start from an experimental study.

We sample several sets of sequences of different lengths (5, 10 and 30 letters). As we did in Sect. 4, Fig. 13 shows the average difference between the estimated and the true probabilities for different types of patterns. We note that the smaller the length of the sequence, the better the estimates. In this context, it seems to be very interesting to have an estimate of the expectation of the length of the sequences to assess the quality of the PDFA.

6.1.2 Expectation of the length of the sequences

Let l be the length of a given sequence accepted by a PDFA (*i.e.* ending in a final state). l is a random variable whose expected value is equal to:

$$E(l) = \sum_{\delta=0}^{\infty} \delta \times P(l = \delta), \tag{38}$$

where $P(l = \delta)$ is the probability of a sequence to have δ letters. Since the PDFA is not learned from the sequences in this case, $P(l = \delta)$ is unknown, but it can be estimated by $P(q_0, l = \delta)$. In Sect. 5.2 we have defined $\tau_{S,T} = \sum_{z,q(S,z)=T} \pi(S, z)$ as the probability to use one transition between states S and T. The probability to have a sequence of δ letters is the probability to use any transition and then to have a sequence of $\delta - 1$ letters. We can establish that:

$$P(S, l = \delta) = \sum_{T \in Q} \tau_{S,T} \times P(T, l = \delta - 1). \tag{39}$$

This is a geometric series of common ratio τ and first term $P(S, l = 0) = \pi_F(S)$. So we get

$$P(l = \delta) = \tau^\delta \times F, \tag{40}$$

where F is the vector of values of $\pi_F(S)$. Using (38) we deduce that:

$$\hat{E}(l) = \sum_{\delta=0}^{\infty} \delta \times \tau^\delta \times F. \tag{41}$$

Let us provide an example to show that (41) gives a correct estimation of $E(l)$. From the set of sequences of Table 2, we deduce that $E(l) = \frac{65}{15} = 4.33$. Applying the formula (see (41)), from the PDFA of Fig. 10, we obtain that $\hat{E}(l) = 4.311$, very close to $E(l)$ despite a small number of sequences ($N = 15$). We now have all the tools to verify if a PDFA is good enough to be mined in a context of privacy preservation. We present in the following a whole system able to extract frequent routes in a town without any information about the car drivers.

6.2 Car flow modeling

In order to bring to the fore the interest of our approach in a context of privacy preservation, we built an application based on road traffic called TRAFFIC MINER. This software allows us to download a given map and model it with a graphical interface in a graph form: one-way and two-way roads as transitions of the PDFA, entry gates, exit gates and crossroads respectively as initial, final and other states. Figure 14 describes an example of use of our software on a map of Arlington, near Washington. On each street, we put a counter to get the number of cars going through it (for example, 316 cars between states 15 and 17). Some of these counters are devoted to compute the number of cars which leave the map (representing final states). When the map is modeled, we can simulate the traffic by generating a random flow of cars. At any moment, we can stop the flow and get a PDFA $A = \langle Q, \Sigma, q, \pi, \pi_I, \pi_F \rangle$ modeling the target distribution of drivers' behavior where:

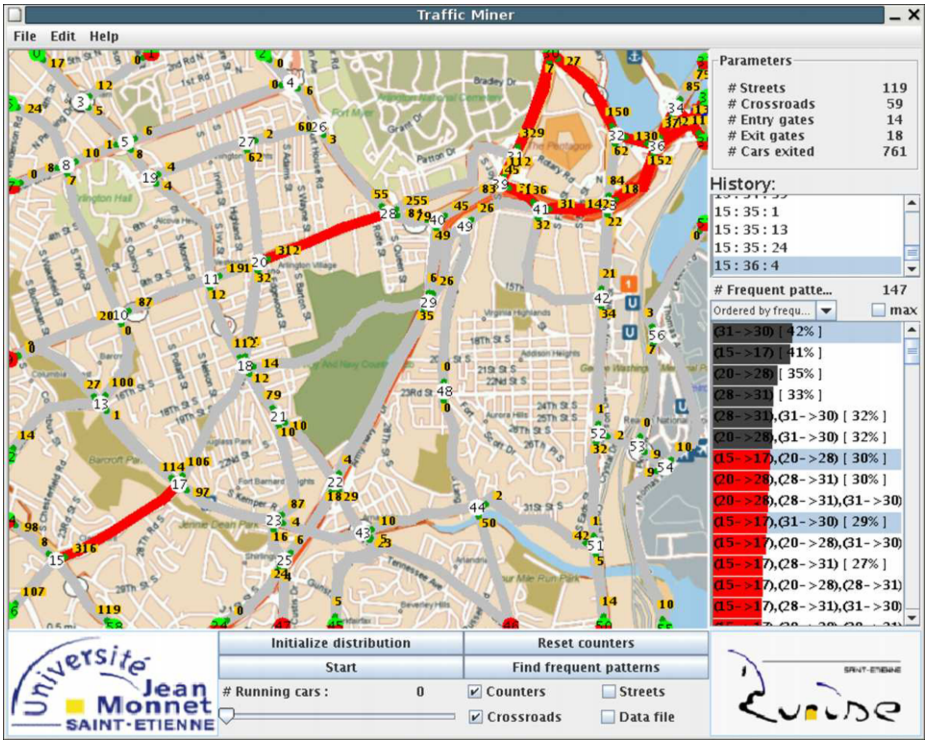
- Q is the set of crossroads, and enter and exit gates,
- Σ is the set of street names,
- q: $Q \times \Sigma \to Q$ defines a transition, *i.e.* a street between two crossroads
- π: $Q \times \Sigma \to [0, 1]$ associates a probability to each pair (S,z), *i.e.* the probability to leave the crossroad S taking the street z,
- π_F: $Q \to [0, 1]$ associates to each final state (*i.e.* the exit gates) a non-null probability $\pi_F(S)$ to leave the map taking the gate S,
- π_I: $Q \to [0, 1]$ associates to each initial state (*i.e.* to entry gates) a non-null probability $\pi_I(S)$.

According to Definition 1, a PDFA must have only one initial state to be deterministic. In our case, despite the fact that we have several initial states (entry gates), the determinism is not challenged. This is because there does not exist two paths, starting from two initial states, that use the same transition (street). To simulate the road traffic, a multinomial distribution is applied on the entry gates, and others are used on each crossroad to simulate the routes.

6.3 Interest of a car flow modeling

From this PDFA, we can run ACSM to extract patterns that may be very interesting in many domains. First, it may be efficiently used in road traffic regulation. By finding frequent paths taken by the same cars, one could locate places in the map which would deserve some modifications (traffic circles, traffic lights, etc.) to make the traffic more fluid. For example, in Fig. 14, we can locate on the top right corner a place where the traffic is very heavy (streets with a dark color). Second, it could be used to simulate a new traffic organization (modification of street directions, creation of new one-way streets, etc.) to avoid traffic jams. Finally, it could be useful in campaign advertising. Note again that a frequent and significant pattern $w = \langle x_1 x_2 \rangle$, extracted with our model, would express that the majority of cars taking the street x_1 will probably also take later the street x_2. This is determined without any information about the individual trail of the drivers. This is the case of the pattern composed of the street between states 15 and 17 and the street between states 20 and 28 in Fig. 14. Our system is able to discover that those who take the first street will probably also later take the second one.

In practice, many reasons can explain the gap in this pattern. Some people can take the street between states 17, 18, 11 and 20 to leave their children at school. Others can prefer to take the street between states 17, 18 and 20 which constitutes the shortest route. Finally, a last category of people can take the street between states 17, 13, 10, 11 and 20 to avoid traffic jams. But at the end, all the drivers meet together in the street between states 20 and 28. This kind of information could for example help an advertising agency to find the best



Parameters

# Streets	119
# Crossroads	59
# Entry gates	14
# Exit gates	18
# Cars exited	761

History:

15 : 35 : 1
15 : 35 : 13
15 : 35 : 24
15 : 36 : 4

Frequent patte... 147

Ordered by frequ... ☐ max

(31->30) [42%]
(15->17) [41%]
(20->28) [35%]
(28->31) [33%]
(28->31),(31->30) [32%]
(20->28),(31->30) [32%]
(15->17),(20->28) [30%]
(20->28),(28->31) [30%]
(20->28),(28->31),(31->30)
(15->17),(31->30) [29%]
(15->17),(20->28),(31->30)
(15->17),(28->31) [27%]
(15->17),(20->28),(28->31)
(15->17),(28->31),(31->30)

Running cars : 0

☑ Counters ☐ Streets
☑ Crossroads ☐ Data file

Fig. 14 Traffic Miner run on a map of Arlington (USA)

strategic position for billboards: either repeating the same message to increase the effect of the advertisement or putting different ones.

6.4 Experimental results

6.4.1 Effect of the constraints

Let us now evaluate the individual effect of our constraints on the number of extracted patterns. Fixing the prefix length $\theta = 0$, the charts of Fig. 15 and Fig. 16 show the effects of the significance constraints. We tested the influence of PROP_TEST without incorporating FISHER_TEST (first chart, fixing $\alpha_2 = 100\%$) and reciprocally (second chart, fixing $\alpha_1 = 100\%$). Of course, we can note that the stronger these constraints, the more the number of patterns decreases. Moreover, we can note that the more the support p_0 increases, the more the relevance constraints become obviously useless. The chart of Fig. 17 shows the influence of the prefix length constraint θ. Here, θ could be useful to extract patterns in the city center (for instance to build pedestrian precincts) that would avoid obtaining patterns at the periphery of the map. We can see that strengthening this constraint (for a given configuration of p_0, α_1 and α_2) leads to extracting a decreasing number of patterns.

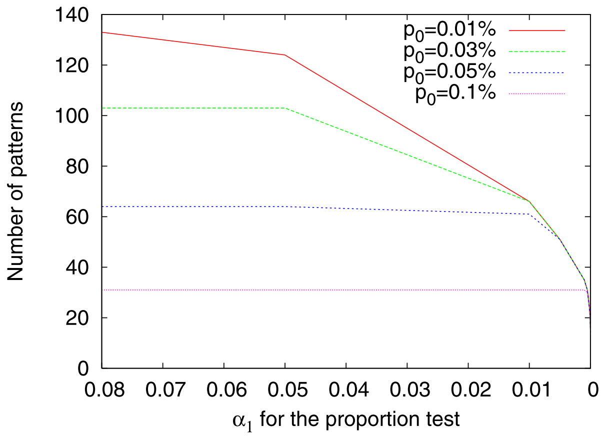


Fig. 15 Effect of the proportion constraint on the number of patterns

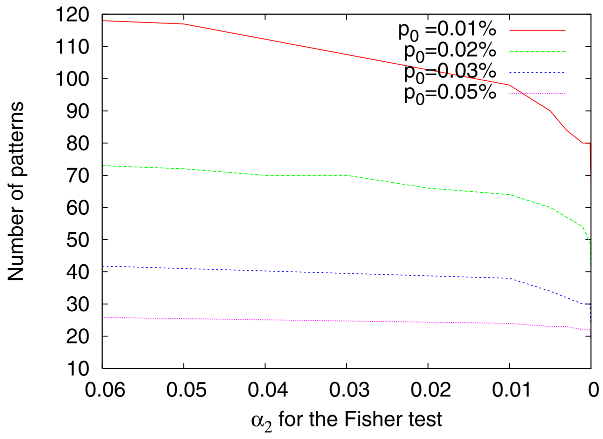


Fig. 16 Effect of the dependence constraint on the number of patterns

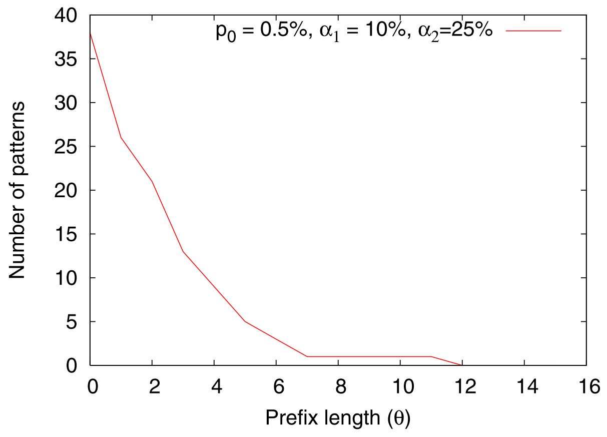


Fig. 17 Effect of Prefix length constraint on the number of patterns

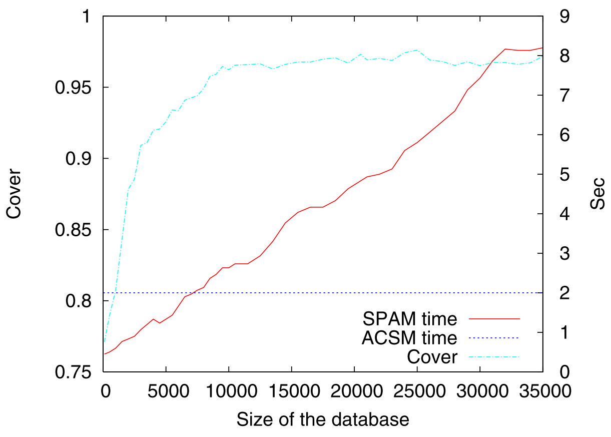


Fig. 18 Comparison of the time complexity between SPAM and ACSM

6.4.2 Experimental comparison with SPAM

Since we do not use the original data, a comparison with other sequence mining algorithms seems more difficult. To overcome this drawback, we sampled sequences from the PDFA and tried to find the same patterns with SPAM. The experimental setup was the following.

We simulated a flow of cars in our map, and we stopped it to get the PDFA. We run ACSM to extract the patterns (here, without constraints to allow us the comparison with SPAM). We measured the time complexity (called ACSM time). Note that we do not take into account the time concerning the construction of the PDFA, since this one is not learned and is directly provided by the application. Directly used on a PDFA, ACSM does not depend on the number of sequences that explains its constant time in Fig. 18. From the PDFA, we sampled many sets of sequences (from 100 to 35000 sequences). From each sample, we ran SPAM to extract a second set of frequent patterns. Both of the algorithms were run with $p_0 = 10\%$. We computed also the time complexity of SPAM by taking into account the sampling time and the mining time (SPAM time). The objective is to determine the minimal size of the set of sequences required to obtain with SPAM the same patterns as ACSM. The chart of Fig. 18 describes the behavior of the two methods. While ACSM has a constant time complexity, the one of SPAM increases with a growing size of the set of sequences. Moreover, since the weighted graph representing the map and the traffic models the target distribution of drivers' behavior, the frequent patterns extracted by ACSM are truly frequent. To study the strength of ACSM with respect to SPAM, we added a curve (Cover) on this chart corresponding to the proportion of true frequent patterns extracted by ACSM (we are not here interested in the false frequent patterns) which have also been extracted by SPAM. We can observe that SPAM needs a large number of sequences to approximate the results of ACSM. We can also note that once the size of the set is sufficiently large, about 10000 examples, almost all the frequent patterns extracted from the PDFA are covered by the frequent patterns founded by SPAM (about 96%). But in this case, the cost of SPAM from a time standpoint is higher than the one of ACSM.

7 Conclusion

In this paper we have shown an original approach to sequence mining, based on the use of a probabilistic automaton. We have given statistical results on the effectiveness of classical and PDFA-based sequence mining algorithms. More precisely, we have provided two bounds that ensure relevant results. We have also introduced statistical and syntactic constraints in order to reduce the search space and achieve specific tasks. We have shown that, in situations that can be modeled as flow control problems, a PDFA-based sequence mining task may preserve privacy thanks to a probabilistic automaton. We have proposed a new constrained sequence mining algorithm (ACSM) based on that data structure and have shown this approach is more efficient than sampling a database from the PDFA and then using a classical sequence mining algorithm. Our algorithm has been implemented in a prototype we have used to visually show the frequent routes of towns without making use of any private information from drivers.

In the future, we plan to use ACSM on two other flow control problems: Web usage mining by using a PDFA to model the structure of a site with the flow of visits, and social network modeling by using a PDFA to model the flow of emails between people. In order to be as efficient as possible, it will be interesting to integrate other constraints such as mingap, maxgap, etc. We also intend to reverse the problem, that is to use information obtained by sequence mining algorithms in order to improve efficiency of grammatical inference algorithms.

References

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the 11th international conference on data engineering* (pp. 3–14). Los Alamitos: IEEE Computer Society.

Agrawal, R., & Srikant, R. (2000). Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD conference on management of data* (pp. 439–450). New York: ACM.

Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the 8th international conference on knowledge discovery and data mining* (pp. 429–435). New York: ACM.

Bayardo, R. J., & Agrawal, R. (2005). Data privacy through optimal k-anonymization. In *Proceedings of the 21st international conference on data engineering* (pp. 217–228). Los Alamitos: IEEE Computer Society.

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: A new and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B*, *57*, 289–300.

Borges, J., & Levene, M. (1998). Mining association rules in hypertext databases. In *Proceedings of the 4th international conference on knowledge discovery and data mining* (pp. 149–153).

Borges, J., & Levene, M. (1999). Data mining of user navigation patterns. In *WEBKDD '99: revised papers from the international workshop on web usage analysis and user profiling* (pp. 92–111). Berlin: Springer.

Borges, J., & Levene, M. (2004). A dynamic clustering-based Markov model for web usage mining. In *CoRR: the computing research repository*. cs.IR/0406032, June 2004.

Callut, J. (2007). *First passage times dynamics in Markov models with applications to HMM: induction, sequence classification and graph mining*. PhD thesis, Université Catholique de Louvain.

Carrasco, R. C., & Oncina, J. (1994). Learning stochastic regular grammars by means of a state merging method. In *Proceedings of the 2nd international colloquium on grammatical inference* (Vol. 862, pp. 139–152). Berlin: Springer.

de la Higuera, C. (2005). A bibliographical study of grammatical inference. *Pattern Recognition*, *38*(9), 1332–1348.

Dupont, P., Denis, F., & Esposito, Y. (2005). Links between probabilistic automata and hidden Markov models: probability distributions, learning models and induction algorithms. *Pattern Recognition*, *38*(9), 1349–1371.

Dupont, P., Callut, J., Dooms, G., Monette, J.-N., & Deville, Y. (2006). *Relevant subgraph extraction from random walks in a graph* (Technical Report 2006-2007). UCL/FSA/INGI, November 2006.

Evfimievski, A. V., Srikant, R., Agrawal, R., & Gehrke, J. (2004). Privacy preserving mining of association rules. *Information Systems*, *29*(4), 343–364.

Fisher, R. A. (1922). On the interpretation of chi-square from the contingency tables, and the calculation of P. *Journal of the Royal Statistical Society*, *85*, 87–94.

Garofalakis, M., Rastogi, R., & Shim, K. (2002). Mining sequential patterns with regular expression constraints. *IEEE Transactions on Knowledge and Data Engineering*, *14*(3), 530–552.

Gionis, A., Mannila, H., Mielikainen, T., & Tsaparas, P. (2006). Assessing data mining results via swap randomization. In *KDD '06: proceedings of the 12th international conference on knowledge discovery and data mining* (pp. 167–176).

Gold, E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, *37*(3), 302–320.

Han, J., Altman, R. B., Kumar, V., Mannila, H., & Pregibon, D. (2002). Emerging scientific applications in data mining. *Communications of the ACM*, *45*(8), 54–58.

Hingston, P. (2002). Using finite state automata for sequence mining. In *Proceedings of the 25th Australasian conference on computer science* (pp. 105–110). Australian Computer Society.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*, 65–70.

Klemettinen, M., Mannila, H., & Toivonen, H. (1999). Interactive exploration of interesting findings in the telecommunication network alarm sequence analyzer. *Information & Software Technology*, *41*(9), 557–567.

Kosala, R., & Blockeel, H. (2000). Web mining research: a survey. *SIGKDD Explorations*, *2*(1), 1–15.

Laur, P., Nock, R., Symphor, J., & Poncelet, P. (2007a). Mining evolving data streams for frequent patterns. *Pattern Recognition*, *40*(2), 492–503.

Laur, P., Symphor, J., Nock, R., & Poncelet, P. (2007b). Statistical supports for mining sequential patterns and improving the incremental update process on data streams. *Intelligent Data Analysis*, *11*(1), 29–47.

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

Megiddo, N., & Srikant, R. (1998). Discovering predictive association rules. In *Knowledge discovery and data mining* (pp. 274–278).

Newton, E. M., Sweeney, L., & Malin, B. (2005). Preserving privacy by de-identifying face images. *IEEE Transactions on Knowledge and Data Engineering*, *17*(2), 232–243.

Pearson, K. (1900). On a criterion that a given system of deviations from the probable in the case of correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *Philosophy Magazine*, *50*, 157–175.

Pei, J., Han, J., & Wang, W. (2002). Mining sequential patterns with constraints in large databases. In *Proceedings of the 11th international conference on information and knowledge management* (pp. 18–25). New York: ACM.

Reber, A. S. (1967). Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, *6*, 855–863.

Shaffer, J. (1995). Multiple hypothesis-testing. *Annual Review of Psychology*, *46*, 561–584.

Spiliopoulou, M., & Pohle, C. (2001). Data mining for measuring and improving the success of web sites. *Data Mining and Knowledge Discovery*, *5*(1–2), 85–114.

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Proceedings of the 5th international conference on extending database technology* (Vol. *1057*, pp. 3–17). Berlin: Springer.

Sweeney, L. (2002). k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, *10*(5), 557–570.

Valiant, L. G. (1984). A theory of the learnable. In *Proceedings of the 16th annual ACM symposium on theory of computing* (pp. 436–445). New York: ACM.

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., & Theodoridis, Y. (2004). State-of-the-art in privacy preserving data mining. *SIGMOD Record*, *33*(1), 50–57.

Webb, G. I. (2007). Discovering significant patterns. *Machine Learning*, *68*(1), 1–33.

Zaki, M. J. (2000). Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the 9th international conference on information and knowledge management* (pp. 422–429). New York: ACM.

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, *42*(1–2), 31–60.