Prediction in wireless networks by Markov chains — Source link

Dimitrios Katsaros, Yannis Manolopoulos

Institutions: University of Thessaly

Published on: 01 Apr 2009 - IEEE Wireless Communications (IEEE)

Topics: Wireless network, Ubiquitous computing, Markov process and Wireless

Prediction in Wireless Networks by Markov Chains

Dimitrios Katsaros[1] Yannis Manolopoulos[2]

[1]Computer & Communication Engineering Department, University of Thessaly, Volos, Greece

[2]Informatics Department, Aristotle University, Thessaloniki, Greece

dkatsar@inf.uth.gr, manolopo@delab.csd.auth.gr

Abstract

Discrete sequence modelling and prediction is an important goal and a challenge for pervasive computing. Mobile client's data request forecasting and location tracking in wireless cellular networks are characteristic application areas of sequence prediction in pervasive computing. This article presents information-theoretic techniques for discrete sequence prediction. It surveys, classifies, and compares the state-of-the-art solutions, suggesting routes for further research by discussing the critical issues and challenges of prediction in wireless networks.

Introduction

The proliferation of cellular networks and the penetration of Internet services are changing many aspects of mobile computing. Constantly increasing mobile client populations utilize diverse mobile devices to access the wireless medium and various heterogeneous applications are being developed to satisfy the eager client requirements. In these environments, seamless and ubiquitous connectivity as well as low client-perceived latencies are two fundamental goals. The first goal calls for smart techniques for determining the current and future location of a mobile, and the second calls for effective techniques for deducing future client requests for "information pieces" (i.e., objects/records from databases, multimedia files, URLs, etc).

Location and request prediction in wireless networks

Location (request) prediction is the task of exploiting the past movements (requests) in deducing what the future locations (requests) will be. Therefore, location (request) prediction can improve the network performance (reduce the user's latency). The ability to determine the mobile client's (future) location can significantly improve the wireless network's overall performance in a number of different ways. Consider for instance the handover procedure in a cellular network covering a metropolitan city; instead of relying on *reactive* approaches, i.e., allocating appropriate resources during the handover, we could come up with *proactive* approaches, i.e., allocating resources before needed, so as to bypass, instead of correct, the negative effect of handover [5]. Additionally, methods like the Shadow Cluster [6], could benefit from location prediction, by refraining from allocating resources to all neighboring cells, but instead, they could allocate resources only to the most probable-to-move cells. Finally, location prediction could be exploited in call admission control techniques and also in sequential paging schemes [1] to reduce the combined paging cost. Apart from the impact upon the network infrastructure, the task of prediction can be employed in improving the performance of many modern data dissemination-based applications (Digital Video Broadcasting-Handheld, services like that offered by the DirectBand Network) which are offored by commercial companies and rely on IEEE 802.11 or 3G wireless networks. In such kind of applications, the clients access the data by monitoring the broadcast channel, until they get the required information, which increases the access latency. Although, client-side caching of frequently accessed data could reduce the problem, it is not a panacea and prefetching can be used to further reduce the access latency.

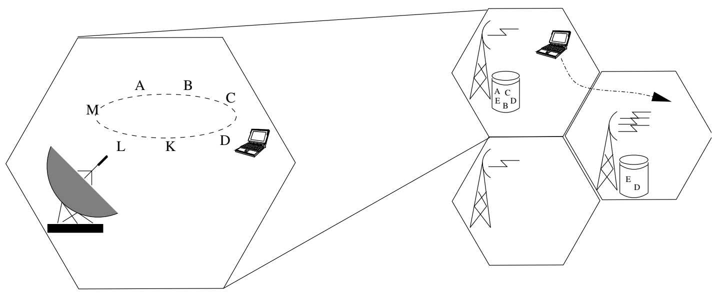


Figure 1. Example mobile transaction. The switching center may allocate more resources and relocate some data (E, D) to the rightmost cell, since the mobile host is expected to move there. In the current cell (zoomed leftmost cell), while the mobile host waits for datum A to arrive in the broadcast channel, may prefetch data C and B, since he forecasted their future use. In many cases, due to the broadcast scheduling algorithm, the data B,C may arrive earlier than A, but their next appearance may be far from A's appearance. Therefore, prefetching reduces the overall latency.

As an example exhibiting the benefits of both location and request prediction, consider the scenario where a mobile client, roaming inside the coverage area of a cellular network, submits transactions to a distributed database system (Figure 1). This database system (by making a simplifying assumption) uses the cellular system's base stations for communicating with the mobile and as its distributed servers as well. To achieve fast response times (i.e., no need for costly remote accesses and expensive handovers) and fault-tolerance (i.e., data availability), a cost-effective solution would be the selective and dynamic allocation of resources (i.e., needed data, bandwidth) in those sites (base stations) which the user would visit soon. Such a scheme would require by the system's side the ability to predict both information needs and the trajectory of the mobile client.

Unifying location and request prediction The issues of location and request prediction had been treated in isolation, but pioneering works ([12] and [1]) are paving the way for treating both problems homogeneously; they exhibited the possibility of using data compression methods in carrying out prediction. The unifying principle is that they model the respective state space as finite alphabets comprised of discrete symbols. In the mobility tracking scenario, the alphabet consists of all possible sites (cells) where the client has ever visited or might visit. In the request prediction scenario, the alphabet consists of all the data objects requested by the client plus the objects that might be requested in the future. Both location and request prediction are related to the ability of the underlying network to record, learn and *predict* the mobile's "behavior". The success of the prediction is presupposed and is boost by the fact that *mobile users exhibit some degree of regularity in their movement and/or in their access patterns* [1]. A "smart" network can record the *movement(request) history* and then construct a *mobility(data access) model* for its clients.

This article provides a unifying framework for all the methods dealing with the location prediction and request forecasting using information-theoretic structures; the framework treats them as *(variable/fixed length) Markov chains* and presents the different families of methods categorizing the state-of-the-art algorithms into their respective families. An important objective of the article is to include in the presentation not only the algorithms which are familiar in the wireless communications community, but also techniques which have been developed in other disciplines, like computational biology, machine learning, and World Wide Web, in order to achieve cross-discipline understanding and proliferation of ideas. The purpose of the categorization is to reveal the shortcomings and advantages of each method and identify routes for further research.

The discrete sequence prediction problem

In quantifying the utility of the past in predicting the future, a formal problem definition is needed. Let Σ be an alphabet, consisting of a finite number of symbols $s_1, s_2, \ldots, s_{|\Sigma|}$, where $|\cdot|$ stands for the length/cardinality of its argument. A *predictor* accumulates sequences of the type $\mathfrak{a}_i = \alpha_i^1, \alpha_i^2, \ldots, \alpha_i^{n_i}$, where $\alpha_i^j \in \Sigma, \quad \forall i, j$ and n_i denotes the number of symbols comprising $\mathfrak{a}_i$. Without lost of generality, we can assume that all the knowledge of the predictor consists of a single sequence $\mathfrak{a} = \alpha^1, \alpha^2, \ldots, \alpha^n$. Based on $\mathfrak{a}$, the predictor's goal is to construct a model that assigns probabilities for any future outcome given "some" past. Using the characterization of the mobility/request model as a stochastic process $(\mathbb{X}_t)_{t \in N}$, we can formulate this goal as follows:

Definition 1 (Discrete Sequence Prediction problem). At any given time instance t (meaning that t symbols $x_t, x_{t-1}, \ldots, x_1$ have appeared, in reverse order) calculate the conditional probability

$$\widetilde{P}[\mathbb{X}_{t+1} = x_{t+1} | \mathbb{X}_t = x_t, \mathbb{X}_{t-1} = x_{t-1}, \ldots],$$

where $x_i \in \Sigma$, $\forall x_{t+1} \in \Sigma$. This model introduces a *stationary Markov chain*, since the probabilities are not time-dependent. The outcome of the predictor is a ranking of the symbols according to their $\widetilde{P}$. The predictors which use such kind of prediction models are termed *Markov predictors*.

The "history" $x_t, x_{t-1}, \ldots$ used in the above definition is called the *context* of the predictor, and it refers to the portion of the past that influences the next outcome. The history's length (also, called the *length* or *memory* or *order* of the Markov predictor) will be denoted by l. Therefore, a predictor which exploits l past symbols, will calculate conditional probabilities of the form:

$$\widetilde{P}[\mathbb{X}_{t+1} = x_{t+1} | \mathbb{X}_t = x_t, \mathbb{X}_{t-1} = x_{t-1}, \ldots, \mathbb{X}_{t-l+1} = x_{t-l+1}]. \quad (1)$$

Some Markov predictors fix, in advance of the model creation, the value of l, presetting it in a constant k, in order to reduce the size and complexity of the prediction model. These predictors, and the respective Markov chains are termed *fixed length Markov chains/predictors* of order k. Therefore, they compute conditional probabilities as above, considering only the events from t up to $t-k+1$, with $k < l$. Although it is a nice model from a probabilistic point of view, these Markov chains are not very appropriate from the estimation point of view. Their main limitation is related to their structural poverty, since there is no means to set an optimized value for k.

Other Markov predictors deviate from the fixed memory assumption, and allow the order of the predictor to be of variable length, i.e., to be a function of the values from the past. These predictors compute conditional probabilities of the form of equation (1), but l is a function of time, i.e., $l = l(x_t, x_{t-1}, \ldots)$.
These predictors are termed *variable length Markov chains*; the length l might range from 1 to t. If $l = l(x_t, x_{t-1}, \ldots) \equiv k$ for all $x_t, x_{t-1}, \ldots$, then we obtain the fixed length Markov chain. The variable length Markov predictors may or may not impose an upper bound on the considered length. The concept of variable memory offers a richness in the prediction model and the ability to adjust itself to the data distribution. Unfortunately, it is not a straightforward problem to choose in a data-driven way the function $l = l(\cdot)$.

The power of Markov predictors

The issue of prediction in wireless networks has received attention during the past years, exploiting techniques like *learning automata*, *Kalman filtering* and *pattern matching*. Learning automata [5] are simple, but they are not considered very efficient learners, because of the need to devise appropriate penatly/reward policies, and due to their slow convergence to the correct actions. Kalman filtering-based methods [7] construct a mobile motion equation relying on specific distributions for its velocity, acceleration and direction of movement. They can not be used for request prediction, but only for location prediction and their performance largely depends on the stabilization time of the Kalman filter and knowledge (or estimation) of the system's parameters. Finally, pattern matching techniques have been used for location prediction [7]. They compile mobility profiles, and perform approximate similarity matching, using the *edit distance*, between the current and the stored trajectories, in order to derive predictions. For the edit distance, it is hard to select the meaningful set of edit operations, to assign weights on them, and so on.

Therefore, why are Markov predictors more appropriate for carrying out location prediction/request prediction, and why this prediction is amenable to Markovian prediction? Their most profound advantage is their generality. They are domain independent and a simple mapping from the "entities" of the investigated domain to an alphabet is all that is required. Thus, they are able to support both location and request prediction. Markovian prediction relies on the *short memory principle*, which says that the (empirical) probability distribution of the next symbol, given the preceding sequence, can be quite accurately approximated by observing no more than the last few symbols in that sequence. This principle fits reasonably and intuitively with how humans are acting when travelling or seeking information. A mobile user usually travels with a specific destination in mind, designing its travel via specific routes (e.g., roads). This "targeted" traveling is far from a random walk assumption, and it is confirmed by studies with real mobility traces [11]. Similarly, almost all request traces exhibit strong spatial locality, which describes correlated sequences of requests.

Families of Markov predictors

Markov predictors create probabilistic models for their input sequence(s) and they use *digital search trees (tries)* to keep track of the contexts of interest, along with some counts used in the calculation of the conditional probabilities $\widetilde{P}$. In the sequel of the paper, we will use the sample sequence of events $\mathfrak{a} = aabacbbabbacbbc$, with *length* equal to $|\mathfrak{a}| = 15$. The *appearance count* of subsequence $\mathfrak{s} = ab$ is $E(\mathfrak{s}) = E(ab) = 2$ and the *normalized appearance count* of $\mathfrak{s}$ is equal to $E(\mathfrak{s})$ divided by the maximum number of (possibly overlapping) occurrences a subsequence of the same length could have, considering the $\mathfrak{a}$'s length, i.e., $E_n(\mathfrak{s}) = \frac{E(\mathfrak{s})}{|\mathfrak{a}|-|\mathfrak{s}|+1}$. The *conditional probability* $\widetilde{P}(b|a)$ of observing a symbol, e.g., b, after a given subsequence, e.g., a, is defined as the number of times that b has shown up right after the subsequence a divided by the total number of times that the subsequence has shown up a all, followed by any symbol. Therefore $\widetilde{P}(b|a) = \frac{E(ab)}{E(a)} = 0.4$.

The Prediction by Partial Match scheme

The Prediction by Partial Match ($\mathcal{PPM}$) is based on the universal compression algorithm reported in [2]. For the construction of the prediction model, it assumes a pre-determined maximal order, say k, for the generated model. Then, for every possible subsequence of length of 1 up to $k+1$, creates or updates the appropriate nodes in the trie. The $\mathcal{PPM}$ predictor for the sequence *aabacbbabbacbbc* is depicted in Figure 2. The maximum context that the $\mathcal{PPM}$ predictor can exploit is k; though, all intermediate contexts with length from 1 to $k-1$ can be used. The interleaving of various length contexts does not mean that this scheme is a variable length Markov predictor, because the decision on the context length is made beforehand and not in a data-driven way.

Apart from this basic scheme a number of variations have been proposed, which attempt to reduce the size of the trie by pruning some of its paths, based on statistical information derived from the input data, e.g., [3]. Apparently, these schemes are offline, making one or multiple passes over the input sequence in order to gather the required statistical information.

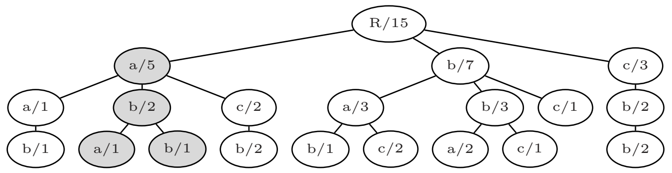


Figure 2. A $\mathcal{PPM}$ Markov predictor for the sequence $aabacbbabbacbbc$.

The Lempel-Ziv-78 scheme

The virtues of the Lempel-Ziv-78 predictor ($\mathcal{LZ}78$) were investigated very early in the literature [1, 12]. The algorithm $\mathcal{LZ}78$ [15] makes no assumptions about the maximal order for the generated model. It parses the input sequence into a number of distinct subsequences, say $\mathfrak{s}_1, \mathfrak{s}_2, \ldots, \mathfrak{s}_x$, such that $\forall j$ $(1 \leq j \leq x)$, the maximal prefix of subsequence $\mathfrak{s}_j$ is equal to some $\mathfrak{s}_i$, for some $1 \leq i < j$. The $\mathcal{LZ}78$ predictor for the sequence $aabacbbabbacbbc$ is depicted in the left part of Figure 3. Though, $\mathcal{LZ}78$ for this example is not able to produce a prediction for the test context ab (i.e., there is no subtree under the gray-shaded node).

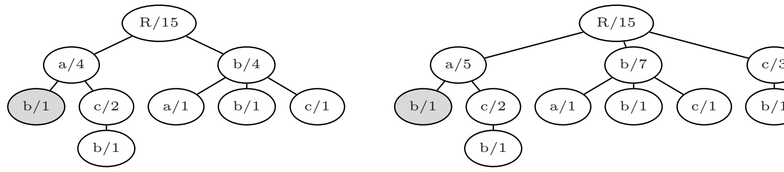


Figure 3. (Left) A $\mathcal{LZ}78$ Markov predictor for the sequence $aabacbbabbacbbc$. (Right) A $\mathcal{LZ}78$ predictor enhanced according to [1, 8].

The $\mathcal{LZ}78$ predictor is an online scheme, it lacks administratively tuned parameters, like lower bounds on appearance counts, and it is a characteristic paradigm of a variable length Markov predictor. Although, strong results do exist which prove its asymptotic optimality and its superiority over any fixed length $\mathcal{PPM}$ predictor, in practice, various experimental studies contradict this result, because of the finite length of the input sequence. The original $\mathcal{LZ}78$ prediction scheme was enhanced in [1, 8] in a way such that apart from a considered subsequence which is going to be inserted into the trie, all its suffixes are inserted, as well (right part of Figure 3).

The Probabilistic Suffix Tree scheme

The Probabilistic Suffix Tree predictor ($\mathcal{PST}$) was introduced in [9] and although it specifies a maximum order for the contexts it will consider, it is actually a variable length Markov predictor and constructs its trie for an input sequence as follows. It uses five administratively set parameters: k, the maximum context length, P_{min}, a minimum normalized appearance count, r, which is a simple measure of the difference between the prediction capability of the subsequence at hand and its direct father node, γ_{min} and α which together define the significance threshold for a conditional appearance of a symbol. Then, for every subsequence of length of 1 up to k, if it has never been encountered before, a new node is added to the trie, provided that a set of three conditions hold. E.g., the subsequence $abcd$ will be inserted into the trie of the $\mathcal{PST}$ iff:

a) $E_n(abcd) \geq P_{min}$, and

b) There exists some symbol, say x, for which the following relations hold:

b$_1$) $\frac{E(abcdx)}{E(abcd)} \geq (1+a)\gamma_{min}$, and

b$_2$) $\frac{\widetilde{P}(x|abcd)}{\widetilde{P}(x|abc)} \geq r$ or $\leq 1/r$, in other words, $\frac{E(abc)}{E(abcd)} * \frac{E(abcdx)}{E(abcx)} \geq r$ or $\leq 1/r$

The $\mathcal{PST}$ predictor with $k = 3, P_{min} = \frac{2}{14}, r = 1.05, \gamma_{mim} = 0.001, \alpha = 0$ for the sequence *aabacbbabbacbbc* is depicted in Figure 4.

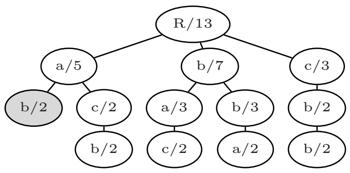
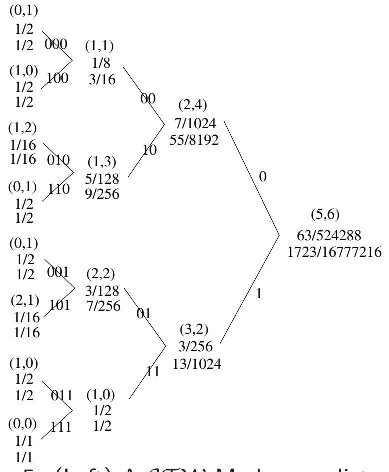
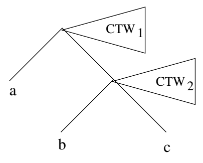


Figure 4. A $\mathcal{PST}$ Markov predictor for the sequence *aabacbbabbacbbc*.

The Context Tree Weighting scheme

The Context Tree Weighting predictor [14] ($\mathcal{CTW}$) is based on the idea of combining exponentially many Markov chains of bounded order, and the original proposition dealt with binary alphabets only. The $\mathcal{CTW}$ assumes a pre-determined maximal order, say k, for the generated model and constructs a *complete binary tree* $\mathcal{T}$ of height k. Each node s maintains two counters a_s and b_s, which count the number of zeros and ones, respectively, that followed context s in the input sequence so far. Additionally, each context (node) s maintains, apart from the pair (a_s, b_s), two probabilities P_e^s and P_w^s. The former, P_e^s, is the Krichevsky-Trofimov estimator for a sequence to have exactly a_s zeros and b_s ones. The latter probability, P_w^s, is the weighted sum of some values of P_e. The $\mathcal{CTW}$ predictor for the sample binary sequence 010|11010100011 is depicted in the left part of Figure 5. With P_e^R and P_w^R denoting the Krichevsky-Trofimov estimate and the $\mathcal{CTW}$ estimate of the root, respectively, we can predict the next symbol with the aid of a $\mathcal{CTW}$ as follows. We make the working hypothesis that the next symbol is a one, and we update the $\mathcal{T}$ accordingly obtaining a new estimate for the root $P_w'^R$. Then, the ratio $\frac{P_w'^R}{P_w^R}$ is the conditional probability that the next symbol is a one.





predictor	sequence it models
CTW_1	aa_a___a__a____
CTW_2	bcbbbbcbbc

Figure 5. (Left) A $\mathcal{CTW}$ Markov predictor for the binary sequence 010|11010100011. (Right) A sketch of the $De\mathcal{CTW}$ Markov predictor for the sequence *aabacbbabbacbbc*.

For the case of non-binary alphabets, Volf [13] proposed the *Decomposed* $\mathcal{CTW}$ ($De\mathcal{CTW}$). Assuming that the symbols belong to a alphabet Σ with cardinality $|\Sigma|$, it considers a full binary tree with $|\Sigma|$ leaves. Each leaf is uniquely associated with a symbol in Σ. Each internal node v defines the binary problem of predicting whether the next symbol is a leaf on v's left subtree or a leaf on v's right subtree, "attaching" a binary $\mathcal{CTW}$ predictor to each internal node (Figure 5).

Comparison of prediction schemes

Implicitly or explicitly all Markov predictors all based on the *short memory principle*, which says that the probability distribution of the next symbol can be approximated by observing no more than the last k symbols in that sequence. Some methods fix in advance the value of k (e.g., $\mathcal{PPM}$, $\mathcal{CTW}$). If the selected value for k is too low, then it will not capture all the dependencies between symbols, degrading its prediction efficiency. On the other hand, if the value of k is too large, then the model will overfit the training sequence. Therefore, variable length Markov predictors (e.g., $\mathcal{LZ}78$, $\mathcal{PST}$) are in general more appropriate from this point of view. This was the motivation for subsequent enhancements to $\mathcal{PPM}$ and $\mathcal{CTW}$ so as to consider unbounded length contexts, e.g., the $\mathcal{PPM}^\star$ algorithm.

On the other hand, variable length predictors face the problem of which sequences and of what length should be considered. $\mathcal{PST}$ attempts to estimate the predictive capability of each subsequence in order to store it in the trie, which results in deploying many tunable parameters. $\mathcal{LZ}78$ employs a prefix-based decorrelation process, which results in some recurrent structures to be excluded from the trie, at least at the first stages. This characteristic is not very important for large sequences, but may incur performance penalty for short sequences; for instance, the pattern *bba* is missing in both variants of $\mathcal{LZ}78$ of Figure 3. Although this example is by no means a kind of proof that $\mathcal{LZ}78$ is inferior to the other algorithms, it is an indication of how an individual algorithm's particularities may affect its prediction performance, especially in short sequences. Despite their superior prediction performance, $\mathcal{PPM}$ schemes are far less commonly applied than algorithms like $\mathcal{LZ}78$, which is favored over $\mathcal{PPM}$ algorithms for its relative efficiency in memory and computational complexity.

Prediction method			**Overheads**			**Particularity**
Family	Variant	Markov class	Train	Parameterization	Storage	
$\mathcal{LZ}78$	[1]	Variable	on-line	moderate	moderate	May miss patterns
	[15]	Variable	on-line	moderate	moderate	
$\mathcal{PPM}$	[2]	Fixed	on-line	moderate	large	Fixed length.
	[3]	Fixed	off-line	heavy	large	High complexity
$\mathcal{PST}$	[9]	Variable	off-line	heavy	low	Parameterization
$\mathcal{CTW}$	[13]	Fixed	on-line	moderate	large	Binary nature
	[14]	Fixed	on-line	moderate	large	

Table 1. Qualitative comparison of discrete sequence prediction models.

From Table 1 we can gain some insights regarding which method is more appropriate for which type of application. To the best of our knowledge, we found no study which compares *all* families mentioned in this article for either the location prediction of the request prediction issue with both synthetic and real data, although a worthwhile study containing comprehensive experiments with real data is reported in [11]. Aiming to provide suggestions for policy selection, we select two primary dimensions; the first dimension reflects the type of the problem (i.e., location or request prediction) and the second dimension reflects the "network part", where the prediction is carried out (i.e., fixed, resource-rich network servers or resource-starving mobile hosts).

For data-consuming applications, some very important intuitive and experimentally confirmed results exist, which state that: a) user request sequences are of considerably varying length, where quite large sequences have significant non-zero probability (Universal Law of Mobile Web Surfing), b) user interests vary significantly with time (not "strong" stationarity), c) many alternative paths exist which lead to the same datum, thus the regularity patterns are "blurred" by noise. Due to the not so strong stationarity, and the existenc of many alternative symbols (in request prediction scenaria), the possibility of using $\mathcal{LZ}78$ types of predictors is not very high. Due to the variance in the length of the individual client's access sequence, the rest of the variable-order Markov predictors are more appropriate; $\mathcal{PST}$ would be a perfect choice under the assumptions that the procedure is performed offline and it runs on a resource-rich server or a laptop.

If energy conservation is the main issue in these applications (e.g., PDAs), then the choice of $\mathcal{PPM}$ style predictors seems more appropriate since they are online, but they sacrifice some prediction performance (due to the relatively small and fixed order model employed) for reduced model complexity. The third observation may turn all prediction methods inefficient, since it violates the "consecutiveness" property of appearance of the symbols in the patterns, upon which property all described Markov predictors rely. In such cases, the modified Markov predictors described in [3] can be employed, but these algorithms are offline and require substantial resources (memory, power) to be executed. Therefore they could only be used by fixed network servers.

Location prediction is considered as a more manageable problem than request prediction, because of the fewer alternatives in possible contexts (i.e., hexagonal architecture of cellular systems, few fixed access points in wireless LANs) and because of the "strong" stationarity (i.e., few habitual routes in campuses/cities, few travel paths in urban regions – road network).

For location prediction applications, several families of Markov predictors could be used in some specific scenarios each. For dynamic tracking of mobile hosts (with the tracking application running either in the network server or the mobile host) $\mathcal{PPM}$ and $\mathcal{LZ}78$ methods are appropriate. The small order $\mathcal{PPM}$ model and the enhanced $\mathcal{LZ}78$ [1] are expected to achieve the best performance, because of the undoubted validity of the stationarity assumption. Indeed, the study in [11] confirmed that intuitive results. These variants are perfect fit for dynamic resource allocation before handovers, as well. For location area design applications, where we are interested in discovering "long-standing" repetitive user routes, the process is offline and therefore methods like $\mathcal{PST}$ or [3] are appropriate and less vulnerable to statistical deviation.

To support the aforementioned design guidelines we performed a performance comparison of the major Markov predictors, as described in [1, 2, 9, 13]. We generated synthetic sequences of 500 symbols each (drawn from a 26-symbol alphabet), falling into four categories: a) with strong stationarity and short (relative to the order of $\mathcal{PPM}$, $De\mathcal{CTW}$) patterns, b) with strong stationarity and longer (relative to the order of $\mathcal{PPM}$, $De\mathcal{CTW}$) patterns, c) with strong stationarity, but high variance in the patterns' length, and d) with piecewise stationarity (cf. Section "Further research"). All patterns were blurred with "white" noise. The average prediction precision of each Markov predictor for each category of sequences is depicted in the four areas of Figure 6.

We can easily see the low performance of the enhanced $\mathcal{LZ}78$ algorithm while the sequences are at their beginings, and how it considerably improves its performance while more symbols are accumulated. $\mathcal{PPM}$ in general performs superior to all its competitors when the patterns' lenght is smaller than its order and this situation is reversed when there is high variance in the patterns' length, in which case $\mathcal{PST}$ prevails. Finally, none of the predictors performs satisfactory for the piecewise stationary sequences.

Further research

The classical result about the duality between lossless compression and prediction implies that any universal lossless compression algorithm can be used to carry out prediction. Although quite a lot of theoretical lossless compression schemes do exist in the literature, only a few of them have been implemented for practical purposes. This is due to the need for effectively combining prediction efficiency, computational complexity and low implementation effort. These three "dimensions" limit the range of possible alternative practical prediction models. Towards this direction, the

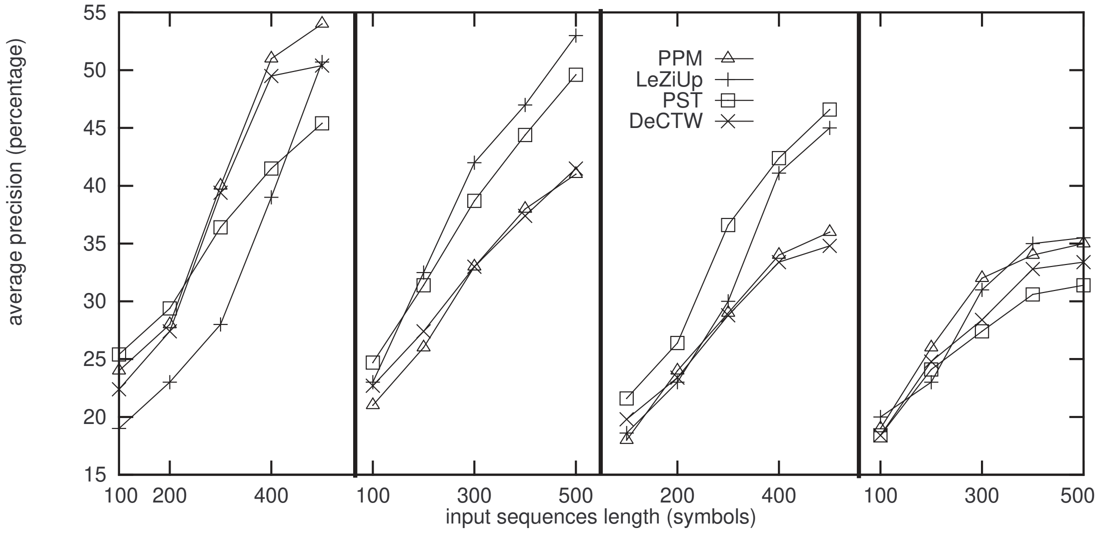


Figure 6. Performance evaluation of major Markov predictors.

Burrows-Wheeler (BW) lossless compression scheme offers significant opportunities for combining the excellent prediction ratios of $\mathcal{PPM}$ and the low complexity of schemes based on $\mathcal{LZ}78$. This compression scheme is composed by three steps: a) the Burrows-Wheeler Transform (BWT), b) a Move-To-Front transform, and c) a variant of Run-Length Encoding and entropy coding. The main difference between BW compression and its context-based competitors ($\mathcal{CTW}$, $\mathcal{PPM}$, $\mathcal{LZ}78$, $\mathcal{PST}$) is that the latter methods encode each symbol knowing the *left* context of its appearance, which helps estimating the probability of the symbol occurrence. In BW schemes, firstly, the *right* context is exploited, and secondly, the information regarding the contexts is lost after the BWT stage. Recovering contextual information could make BW appropriate for prediction in wireless settings, but unfortunately no full solution to this problem has been described yet. So far, no practical prediction scheme is based on the BW scheme.

The cornerstone for building the Markov predictors described in this paper is the "stationarity assumption", which implied time-homogeneous transition probabilities. Under this assumption, the trie of each predictor grows node-by-node increasing the respective node counters, i.e., identical subsequences are "aggregated" (mapped) into the same node of the trie. Under the stationarity assumption, Markovian prediction is well understood. By completely removing the notion of stationarity, it is obvious that we can hardly have any prediction capability, since the fundamental tenet of predictability is violated, i.e., some degree of ergodicity.

In-between these two extremes lies the more realistic case of *piecewise stationarity*, where the sequence of symbols is treated now as a series of non-overlapping *segments*, each of them having been generated by a stationary source. It is supposed that the system has no knowledge about the number of these segments, their duration, the kind of change between them (abrupt or slowly varying). Clearly, the removal of the full stationarity assumption, turns the procedure for creation of predictors mentioned in the previous paragraph, not appropriate at all or of reduced value. For instance, in the simplest case of a mobile client whose roaming patterns change gradually, the predictors will tend to favor the "old habits" of the client and will adapt to the changing conditions in a very slow rate. Therefore, the assumption of non time-homogeneous transition probabilities turns the current predictors inefficient and raise some design challenges for any new scheme that will be designed to address this assumption. Although research works exist dealing with piecewise stationarity (e.g., [10]) these works mainly focus on memoryless sources and have not considered Markov sources.

Markov predictors could still achieve prediction efficiency for piecewise stationary sources, but they should be armed with some smart mechanisms. For instance, an enhanced $\mathcal{LZ}78$ scheme is presented in [1, 8], which inserts into the trie all proper suffixes of a discovered subsequence (a $\mathcal{LZ}78$ word). Indeed, this improvement is shown to outperform the classic $\mathcal{LZ}78$ decorrelation scheme. Although not explicitly mentioned as such in [1], this enhancement can be seen as a method of modeling intra-word correlations, i.e., an approach to account for the fact that full stationarity might not be valid for the sequence, in which case the original decorrelation procedure finds the wrong words. This technique seems appropriate in cases of abrupt changes between stationary segments, but does it really help for piecewise stationary slowly varying sequences? Also, the enhancement reported in [4] takes a first step toward addressing this issue, but in many real applications, for instance in daily movement patterns of mobiles, it may be the case that the stationary segments are repeating. How could we discover (segment the sequence) and exploit such repeating statistics? As we mentioned, "full aggregation" (i.e., accumulating counts in nodes) is not helpful; partial (controlled) or no aggregation could be considered as well, but in any case novel prediction algorithms should be designed, since, as [11] concluded, there is significant gap between the performance of the examined Markov predictors and an "optimal" offline predictor.

Summary

Designers of modern wireless networks are increasingly confronted with the issues of providing seamless and ubiquitous connectivity in a system-independent nature, as well as providing low access latency to data-hungry applications. Smart wireless networks could deduce future client locations and allocate resources in advance, in order to mitigate the negative effect of handovers and perform paging in a cost-effective manner. Similarly, smart agents running on the mobile clients could forecast future client data needs and preload the respective data from the broadcast channels, while waiting to get the explicitly-requested information pieces.

This article presented the issues of location and request prediction in wireless networks in a homogeneous fashion, characterizing them a discrete sequence prediction problems, and surveyed the major Markovian prediction methods. The article by no means serves as an exhaustive survey, but as a vehicle to promote understanding and proliferation of ideas. We explained the virtues of Markov predictors and showed important yet to be addressed research issues. We envision predictive model design as a fertile research area.

Acknowledgements

Research supported by a *ΓΓΕΤ* grant in the context of the project "Data Management in Mobile Ad Hoc Networks" funded by *ΠΥΘΑΓΟΡΑΣ ΙΙ* national research program.
The authors wish to thank the editor and the three anonymous reviewers for their valuable comments.

References

[1] A. Bhattacharya and S. K. Das. LeZi-Update: An information-theoretic framework for personal mobility tracking in PCS networks. *ACM/Kluwer Wireless Networks*, 8(2–3):121–135, 2002.

[2] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.

[3] M. Deshpande and G. Karypis. Selective Markov models for predicting Web page accesses. *ACM Transactions on Internet Technology*, 4(2):163–184, 2004.

[4] K. Gopalratnam and D. J. Cook. Online sequential prediction via incremental parsing: The Active LeZi algorithm. *IEEE Intelligent Systems*, 22(1):52–58, 2007.

[5] M. Kyriakakos, N. Frangiadakis, L. Merakos, and S. Hadjiefthymiades. Enhanced path prediction for network resource management in wireless LANs. *IEEE Wireless Communications*, 10(6):62–69, 2003.

[6] D. A. Levine, I. F. Akyildiz, and M. Naghshineh. A resource estimation and call admission algorithm for wireless multimedia networks using the shadow cluster concept. *IEEE/ACM Transactions on Networking*, 5(1):1–12, 1997.

[7] T. Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. *IEEE Journal on Selected Areas in Communications*, 16(6):922–936, 1998.

[8] A. Misra, A. Roy, and S. K. Das. An information-theoretic framework for optimal location tracking in multi-system $4G$ wireless networks. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 1, pages 286–297, 2004.

[9] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25(2–3):117–149, 1996.

[10] G. I. Shamir and D. J. Costello Jr. Asymptotically optimal low complexity sequential lossless coding for piecewise-stationary memoryless sources – Part I: The regular case. *IEEE Transactions on Information Theory*, 46(7):2444–2467, 2000.

[11] L. Song, D Kotz, R. Jain, and X. He. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, volume 2, pages 1414–1424, 2004.

[12] J. S. Vitter and P. Krishnan. Optimal prefetching via data compression. *Journal of the ACM*, 43(5):771–793, 1996.

[13] P. Volf. *Weighting techniques in data compression: Theory and algorithms*. PhD thesis, Technische Universiteit Eindhoven, 2002.

[14] F. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, 41(3):653–664, 1995.

[15] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.