Sequence Prediction with Sparse Distributed Hyperdimensional Coding Applied to the Analysis of Mobile Phone Use Patterns

Okko J. Räsänen and Jukka P. Saarinen

Abstract—**Modeling and prediction of temporal sequences is central to many signal processing and machine learning applications. Prediction based on sequence history is typically performed using parametric models, such as fixed-order Markov chains (n-grams), approximations of high-order Markov processes, such as mixed-order Markov models or mixtures of lagged bi-gram models, or with other machine learning techniques. This paper presents a method for sequence prediction based on sparse hyperdimensional coding of the sequence structure and describes how higher-order temporal structures can be utilized in sparse coding in a balanced manner. The method is purely incremental, allowing real-time on-line learning and prediction with limited computational resources. Experiments with prediction of mobile phone use patterns, including the prediction of the next launched application, the next GPS location of the user, and the next artist played with the phone media player, reveal that the proposed method is able to capture the relevant variable-order structure from the sequences. A comparison with n-grams and mixed-order Markov models shows that the sparse hyperdimensional predictor clearly outperforms its peers in terms of unweighted average recall and achieves an equal level of weighted average recall as the mixed-order Markov chain but without the batch training of the mixed-order model.**

Index Terms—**machine learning, real-time systems, prediction methods, sequences, time series analysis**

I. Introduction

This work concentrates on the prediction of discrete (categorical) sequences on computational platforms where iterative training is not feasible due to limitations in data storage or computational power. For instance, prediction of user behavior from mobile phone sensors and states is a task with a large amount of potentially useful data, but with only limited possibilities for pre-training of the predictive models due to user-specific differences in the behavior. By using incrementally learning predictive algorithms that scale up to potentially large and a priori unknown representational spaces while using finite pre-allocated computational resources, the models could be learned on the fly as more and more data is observed. Moreover, one-shot learning from individual episodic instances would be desirable if the amount of training data is very low.

In order to accomplish the prediction problem, this work proposes a method based on sparse hyperdimensional coding [1] of a sequence structure. The idea is to represent the previously observed history of each possible sequence state using a single vector of a very large dimensionality – a sort of rich prototype for the preceding context of the state. In this vector, the typical preceding states and their precise temporal ordering are represented in a compositional manner. The prediction of the next most likely state is achieved by building a similar compositional vector representation from the currently observed sequence and measuring the similarity of this vector to the previously learned context vectors of each possible future state. The state corresponding to the most similar context vector is then chosen as the hypothesis for the next sequence state.

This type of compositional coding provides a natural means of representing variable order structure in the data, i.e., the prediction is based on the overall similarity of the situation at different distances in the sequence history instead of using fixed conditional probabilities based on precisely observed preceding states, as in standard Markov processes. If the current sequence has only partial similarity to the previously learned structure, only the similar parts are automatically used in the prediction process. In comparison to earlier work on hyperdimensional predictive methods [2,3], the current work describes a non-parametric mutual-information based method for weighting observations at different temporal distances in the compositional representation, leading to enhanced performance in the prediction process for sources that do not follow first-order Markov statistics.

The proposed sparse distributed predictor (SDP) is evaluated by using it to predict real life mobile phone user data [4] including application launch logs, sequences of GPS locations, and music playback logs. The experiments show that the SDP clearly outperforms all studied Markov-process

Manuscript received August 6, 2014. Revised version received April 22, 2015. This research was funded by Tekes and Tivit joint program Data to Intelligence (D2I) and by Academy of Finland.

O. Räsänen is with the Dept. Signal Processing and Acoustics, Aalto University, P.O. Box 13000, FI-00076 Aalto, Finland (phone: +358-50-441-9511; fax: +358-9-460 224; e-mail: okko.rasanen@aalto.fi).

Jukka P. Saarinen is with the Nokia Technologies, P.O. Box 1000, FI-33721 Tampere, Finland (e-mail: jukka.p.saarinen@nokia.com).

baseline methods in the tasks when measured in terms of unweighted average recall, and obtains an equal level of performance with the mixed-order Markov model in weighted average recall but with purely incremental (non-iterative) training. However, in comparison to more powerful state-of-the-art methods such as long short-term memory (LSTM) networks [5] or their deep variants [6], the present method is unable to efficiently solve some of the more complex long time lag problems (see [5] for examples). This is the cost associated with purely incremental training and the absence of sensitive hyperparameters that would have significant impact on the system performance across different prediction tasks.

The paper is organized as follows. First, a brief introduction to the standard solutions in sequence prediction is given in section I.A, followed by a review of previous research on sparse hyperdimensional coding in sequence modeling. Section II describes the SDP predictor. Section III describes the experimental setup, the baseline methods, and the results from the prediction experiments. Finally, a discussion and conclusions are given in the final section.

A. Prior art

Although numerous learning algorithms can be used for discrete categorical prediction, a comprehensive review of the existing work is way beyond the scope of this paper. Instead, the aim here is to provide a number of examples of how the dependencies at different temporal distances in sequences have been utilized in the prediction task in the context of Markov models and how sparse distributed coding has been applied so far to the coding of sequential structures.

In discrete sequence prediction, the sequences may originate from a multivariate time series that are vector quantized into a discrete form through clustering or state-space partitioning, or it may be categorical in nature (e.g., genomic sequences or words of a language). The basic problem is to infer the most likely next state w_{t+1} of a discrete sequence $X = \{w_1, w_2, \ldots, w_t\}$, where w_t is the most recently observed state and all states belong to a finite set of A unique states ($w_i \in \{1, 2, \ldots, A\}$). Typical solutions assume that the next state is only dependent on m previous observations $X_{\text{context}} = \{w_{t-m+1}, w_{t-m+2}, \ldots, w_t\}$, corresponding to a Markov process of the same order. On this basis, the most straightforward approach is to model the next state as a discrete conditional distribution, or n-gram (with $n = m+1$), of the form

$$P(w_{t+1} \mid w_t, w_{t-1}, \ldots, w_{t-m+1}) \tag{1}$$

A maximum-likelihood solution for the probabilities can be derived directly from the counts of state joint-occurrences in the data. Although efficient for many problems such as language models in speech recognition [7], the n-grams have two inherent problems: 1) the order of the n-gram should ideally correspond to the order of the process generating the data or otherwise information is lost and 2) the number of parameters in an n-gram model grows exponentially as a function of the n-gram order. In practice, the n-grams that can be reliably estimated from typical finite data sets are limited to relatively low orders. In addition, the underlying source generating the data may not be a truly fixed-order Markov process, but the optimal n-gram length may vary across sequence position, making fixed-order model an inaccurate descriptor of the process.

In order to overcome the limitations of the standard n-grams, numerous improvements have been proposed to capture the long-distance temporal dependencies in sequences. In the so-called back-off models [8], n-grams of various orders are first estimated from the training data. During the prediction (or sequence probability estimation), when faced with a rarely observed n-gram of a high order, the probability of the n-gram is interpolated from lower order n-grams whose probabilities are known more reliably. Although successful in combining information from n-grams of different orders, the smoothing process used in the back-off models may cause loss of information regarding the sequence structure [9].

Mixed-order Markov chains [9] have been proposed to combat the problem of smoothing in the back-off models. In the mixed-order Markov model, the n-grams are replaced by a set of skip-k transition matrices (bigrams) $M(w_{t-k}, w_{t+1})$ so that the probability of a state w_{t+1} is computed as

$$P(w_{t+1} \mid w_t, \ldots, w_{t-m+1}) = \sum_{k=0}^{m-1} \lambda_k(w_{t-k}) M_k(w_{t-k}, w_{t+1}) \ldots \prod_{j=0}^{k-1} [1 - \lambda_j(w_{t-j})] \tag{2}$$

i.e., as a set of predictions from a number of different temporal lags k, where the lag-specific predictions depend on the transition parameters $M_k(w_1, w_2)$ and the mixing weights $\lambda_k(w)$. Both the transition matrices and the mixing weights are iteratively estimated using the EM algorithm. The parameters M_k can be initialized from the raw frequency-based lagged bigrams that already provide a solid starting point for the model (see [9] for details). When performed in this manner, the prediction is no longer dependent on an exact match in the sequence history up to the specified model order, but partial similarity is sufficient to provide finite probability estimates for the next state. Saul & Pereira [9] show that the mixed-order model clearly outperforms the standard back-off n-gram model [8] in the context of language models for speech recognition. This is mainly due to its ability to provide a finite probability for any n-gram whose one or more bi-gram components have occurred in the training data and due to the ability to utilize long-distance dependencies in the data when they exist. Also, the number of parameters in the mixed-order model increases only linearly as a function of model order in comparison to the exponential growth in the standard n-grams. The drawback of the mixed-order model is that the training has to be performed iteratively across the entire training data, requiring storage of the entire sequence history and making incremental updates to the model complicated.

Mixed-order temporal structure is also captured in the so-called *mixture transition distributions* (MTD) [10], where the formulation in Eq. (1) is represented in the form

$$P(w_{t+1} \mid w_t, \ldots, w_{t-m+1}) = \sum_{k=0}^{m-1} \lambda_k M(w_{t-k}, w_{t+1}) \tag{3}$$

with the constraint $\sum\lambda_k = 1$. In other words, there is only one transition matrix $M(w_1,w_2)$ and a set of lag-specific weights λ_k, and these parameters are optimized using constrained non-linear optimization. Due to the parsimony of the MTD model (only $m+A^2$ parameters in comparison to the $(A\text{-}1)A^m$ parameters of standard n-grams), the model is fitted to data with small number of training samples or high state-space dimensionality. Berchold & Raftery [11] show that the model can outperform standard Markov chains in modeling wind direction and epileptic seizure data. Prinzie & Van den Poel [12] show similar results for the modeling of consumer purchasing sequence patterns.

In the generalized MTD (GMTD) [11], the requirement for a single transition matrix is relaxed by having a different transition matrix for each lag, leading to the form

$$P(w_{t+1} \mid w_t,...,w_{t-m+1}) = \sum_{k=0}^{m-1} \lambda_k M_k(w_{t-k}, w_{t+1}) \qquad (4)$$

in which estimation of λ_k and $M_k(w_1,w_2)$ can be performed with the EM algorithm or other iterative optimization methods (see [11]). This additional degree of freedom leads to further improvements in the prediction power on complex data as long as there is a sufficient amount of training data available for reliable estimation of the model parameters.

Finally, in variable length Markov chains (VLMC) [13,14] the assumption is that the order of the Markov process changes across position in the data. Therefore the model order itself can be modeled as a function of the sequence history $m = f(w_t, w_{t-1}, w_{t-2},\ldots)$, and the higher-order parameter estimates are grouped together when their predictions of the future state are equal, leading to increased accuracy of the parameters. In [14], an efficient algorithm to estimate the variable length model is given. The authors also show the usefulness of capturing the variable order structure with the VLMC in the analysis of recurring structures from DNA sequences.

In the context of the present work, the main issue with the higher-order Markov chain approximations described above is that they typically require iterative estimation of the model parameters. In some domains such as the presently investigated prediction of mobile phone use patterns, storage of all relevant training data may not be feasible. Therefore a purely incremental system would be beneficial.

B. Sequence prediction with hyperdimensional vectors

Incremental sequence prediction has already been previously studied in the context of hyperdimensional computing (HC). The idea in HC is to represent processed entities (symbols, values, objects, states) as random sparse vectors having a huge dimensionality, typically counted in thousands (e.g., $d = 10000$). Each represented element, say, a sequence state w_i (e.g., a word of a language or GPS location of a phone user), is coded by a unique randomly generated hyperdimensional vector $\mathbf{v}_i$, from now on referred to as a *hypervector*. The values of the vector can all be randomly assigned (e.g., being +1 or -1 for a binary vector), or the vectors can be ternary with only a small number of non-zero ±1 elements (e.g., 5% of all values) at randomly assigned positions [15, 16]. Moreover, distances between the vectors can be measured using a variety of metrics, typical metrics being either Hamming distance (for binary data) or dot product of the vectors. In all variants, the large dimensionality of such vectors leads to a number of interesting properties: First, the representations are highly robust against distortions and noise in the coding process due to the distribution of information across the entire vector length. Secondly, the distribution of the mutual distances between all possible random vectors is tightly packed around the mean of the distances. This means that the probability that a distance between any two randomly drawn vectors is notably smaller than the average distance in the vector space is extremely small[1]. This property of near orthogonality of random vectors leads to the practical property that the sparse coding can be used to represent sets of items as vector addition of the hypervectors of the items in each set (i.e., $\mathbf{v}_{set} = \mathbf{v}_{state1}+\mathbf{v}_{state2}$) without an increase in the dimensionality of the representation. Importantly, the combined set representation is similar to its components in the hyperspace, and therefore the individual items can still be recovered from the holistic representation (see [1]). Moreover, the similarity metrics of the vectors are maintained so that two sets with similar sub-components are similar to each other if their components are similar to each other. This applies even if the overall number of sub-components differs between the vectors, providing a continuous measure of similarity (e.g., in terms of the dot product) that takes into account only those parts of the representation that are present in both vectors (so-called partial matching). Naturally, sets containing qualitatively different types of tokens (e.g., spoken words and visual objects) and sets of sets can all be combined into new hypervectors in a similar manner and the chosen metric is always applicable between the representations. Also, if required by the computational architecture, the sum vector can be rounded back to a binary or ternary vector without losing the basic functionality of the sum coding, although at the cost of coding efficiency. Since the early work of Kanerva [17], HC has been applied to numerous domains including, e.g., visual character recognition [18], cognitive software agents [19], speech recognition [20,21], robotics [22], pattern denoising [23], and naturally, sequence prediction [2,3,17].

In the earliest work on sequence modeling utilizing HC, Kanerva proposed a special memory architecture named sparse distributed memory (SDM) for learning of associations between hyperdimensional representations [17,24]. The basic idea in SDM is to use one input vector $\mathbf{v}_i$ as a memory address that activates a small fraction of M hard-coded memory locations in the hyperspace sufficiently close to the given address (typically $M >> 10000$ when using pre-allocated addresses). Another input vector (*content vector)* $\mathbf{v}_j$ is then

[1] Kanerva [1] gives an example with 10000–dimensional binary vectors: given any data point in the 10000–dimensional binary space, all other possible data points are, on average, 5000 bits away in the Hamming space. However, less than a thousand-millionth of the data points are closer than 4700 bits. This means that a vector with almost half of its bits randomized is still distinct, in practice, from any randomly drawn vector.

stored to each of these locations by using the vector addition principle. During recall, an address vector again activates a number of nearby memory slots and the memory output is obtained by taking the averaged result across all the activated slots. The motivation for SDM, together with the general benefits of HC discussed above, is its content addressability: representations can be retrieved from the memory based on only partial similarity, providing an inherent mechanism for generalization towards similar but not identical situations.

In order to learn a sequential structure using the SDM, the memory can be operated in a heteroassociative learning mode where hypervectors corresponding to the preceding sequence states are associated to the currently observed state. For example, in order to learn a first-order sequential structure, the input vector $\mathbf{v}_{t+1}$ can correspond to a sequence state w_{t+1} while the address vector $\mathbf{v}_t$ is derived from the previous state w_t. In so-called k-folded learning, a separate SDM is trained for associations at each temporal lag $\mathbf{v}_{t-k} \rightarrow \mathbf{v}_t$ in order to utilize higher-order temporal information in the prediction [17]. This type of learning has been studied, e.g., in the modeling of service robot movement trajectories [22,26].

Due to the additivity principle of the vectors, the context of multiple preceding states can also be coded as a single vector $\mathbf{v}_{\text{context}} = \mathbf{v}_t + \mathbf{v}_{t-1} + \ldots + \mathbf{v}_{t-m+1}$, possibly with each $\mathbf{v}_{t-k}$ uniquely determined not only by the state identity, but also by the relative position (lag) of the state with respect to current time (e.g., as in a shift register; see [2]). However, the problem in both k-folded learning and in the single combined context vector is that the preceding states are not all equally important in the prediction task since the predictive value of a state typically diminishes as the distance between the states in the sequence increases. When the distance metrics are applied between sparse representations, the overall scale of the codes from the different lags will determine how much weight is given to each sub-component in the combined context vector. The problem is to a find a weighted representation of the history that can be associated with the next state $\mathbf{v}_{t+1}$ so that the prediction accuracy becomes maximal (see also Fig. 1):

$$\mathbf{v}_{\text{context}} = \lambda_0\mathbf{v}_t + \lambda_1\mathbf{v}_{t-1} + \ldots + \lambda_m\mathbf{v}_{t-m+1} \qquad (5)$$

Bose et al. [2] attempted to solve the weighting problem in SDM by setting geometrically diminishing weights to the history state codes. In their approach, the most recent state was represented with weight $\lambda_0 = 1$ and the preceding states as $\lambda_k = b^k$, where b is a constant weight. Their results with artificial symbol sequences showed that the use of lag-dependent (shift register) coding of the previous states together with diminishing history weights leads to a better sequence prediction performance than using either of them alone. Snaider & Franklin [3] have also proposed the use of SDM for sequence prediction with geometrically diminishing weighting of the preceding states so that the weight coefficient is defined manually. The problem in both of these approaches is that the assumption of a geometric decay is arbitrary with respect to the real dependency of states at different temporal

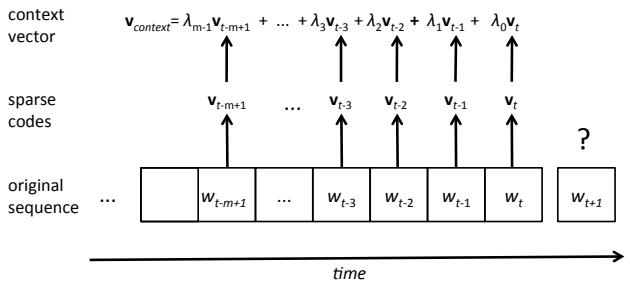

Fig. 1. An example of the sparse coding process. Each history state w_{t-k} of a discrete sequence X (bottom) is mapped onto a hyperdimensional sparse vector $\mathbf{v}_{t-k}$ that is orthogonal with all the other sparse codes used. The overall history up to m previous elements is represented as a weighted sum $\mathbf{v}_{\text{context}}$ of the corresponding sparse codes.

distances and can only apply to the first-order Markov processes if the parameter b is properly set. Neither Bose et al. [2] or Snaider & Franklin [3] tested their systems with non-artificial data of an unknown Markov order, and therefore the scalability of the approaches remains unclear.

In addition to the absence of a satisfactory solution to the problem of history weighting, a major challenge in the practical application of the standard SDM is its computational complexity ($O\{NM\}$ for accessing hard locations) and the problem of generating an optimal set of hard locations to the address matrix $\mathbf{A}$ if the data is not uniformly and randomly distributed across the entire space (see, e.g., [27,28] for possible solutions to the address problem). These challenges make the application of full-scale SDM non-trivial even with modern computers, not to mention mobile platforms with limited computational resources.

However, SDM is not the only architecture that can utilize the benefits of HC. Random indexing (RI) [15,16] uses HC-based representations to estimate the degree synonymy between words in text (Fig. 2). For a sequence of words $X = \{w_1, w_2, \ldots, w_L\}$ (e.g., a text corpus), the left context of each word w_t is defined as $\{w_{t-m}, \ldots, w_{t-1}\}$ and the right context as $\{w_{t+1},\ldots,w_{t+m}\}$. For each unique word w_i in the sequence, there is a sparse hyperdimensional random vector $\mathbf{v}_i$ associated with the word. Therefore, the left and right contexts of each word can be described as $\mathbf{v}_{\text{context}} = \mathbf{v}_{\text{left}} + \mathbf{v}_{\text{right}} = \mathbf{v}_{w(t-m)}+\ldots+\mathbf{v}_{w(t-1)} + \mathbf{v}_{w(t+1)}+\ldots+\mathbf{v}_{w(t+m)}$ (cf. Eq. (5)). The core of the RI is a memory matrix $\mathbf{H}_{A\text{x}d}$ with a unique row for each possible word w_i and the number of columns equal to the dimensionality of the hyperspace. In the beginning, $\mathbf{H}$ is initialized with all entries zero. For each occurrence of word w_i in the training data, the corresponding context vector $\mathbf{v}_{\text{context}}$ is computed and summed to the w_ith row of $\mathbf{H}$. As a result, the rows of $\mathbf{H}$ become descriptions of the typical contexts in which each word w_j occurs. By normalizing the rows of $\mathbf{H}$ into unit vectors and computing $\mathbf{S} = \mathbf{H}\mathbf{H}^{\text{T}}$, a measure of the pair-wise synonymy of words $S(w_i, w_j)$ is obtained since the words with a similar meaning occur in similar contexts [15]. In comparison to latent semantic analysis (LSA) [29], RI allows fast word synonymy estimation without having to compute the singular value decomposition of a huge co-occurrence matrix. In the standard synonymy estimation with LSA, the word-context

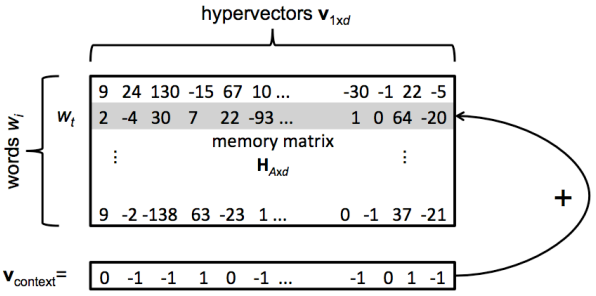

Fig. 2. Schematic view of the standard RI [15]. The temporal context of each word w_t is coded with a hypervector $\mathbf{v}_{\text{context}}$ that represents the neighboring states $w_{t-m}\ldots w_{t-1}$ and $w_{t+1}\ldots w_{t+m}$. For all w_t in the data set ($t \in [m, L\text{-}m]$), the corresponding context vectors are accumulated in the rows of $\mathbf{H}$ corresponding to each w_t. As a result, each row represents the sum of contexts in which each possible word w_i occurs. Contextual similarity (synonymy) of a word pair $\{w_i, w_j\}$ is derived by normalizing the rows of $\mathbf{H}$ and computing the distance between the rows i and j using a chosen metric. In the web browsing predictor of [30], the context vector is formed from a set of contextual variables such as time, location, and previous web pages whereas each row represents a web page URL w_i.

co-occurrence matrix grows exponentially as a function of vocabulary size as the context of each word is defined in terms of other words. In RI, on the other hand, dimension of the context representation never increases from the original value.

In [30], the RI-based computation is used to predict the most likely web page that a mobile phone browser user wants to visit next. Instead of explicitly modeling browsing as sequential behavior, the predictor simply combines different sources of information (e.g., current location, time, previous pages, and calendar) into a single hypervector $\mathbf{v}_{\text{context}}$ without any weighting procedure (a bag-of-words representation) and models the occurrence of these context vectors during each visited URL by assigning a unique row w_i in the $\mathbf{H}$ matrix for each URL. Similarly to the predictive work with the SDM, the major shortcoming of approach in [30] is the lack of a suitable weighting scheme that accounts for the varying importance of different information sources, and therefore effective utilization of the fine-grained temporal information from the preceding system states is not possible.

In the current work, we extend the basic idea of hypervectors and RI in order to predict variable-order sequential data. More specifically, we use RI to model the history of preceding contexts of each possible state w_i in a sequence and provide an information-theoretically motivated weighting scheme for the optimal usage of information in the preceding signal states. The proposed method is described in the following section in more detail.

II. Sparse Distributed Predictor

A. *Coding temporal context with mutual information weighted hypervectors*

In order to represent and predict the varying-order structure of sequences, hyperdimensional coding of states with mutual information-based weighting is utilized. The benefit of using hyperdimensional codes is that typical processed data is likely to be sparse, i.e., only a small fraction of all possible

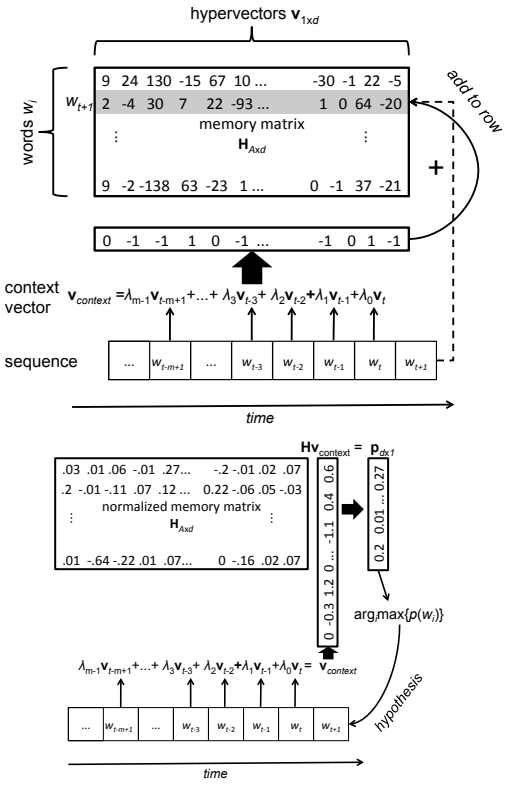

Fig. 3. A schematic view of the SDP. Top: All preceding contexts of each sequence state w_i are coded into hyperdimensional context vectors $\mathbf{v}_{\text{context}}$ and accumulated in the corresponding rows of $\mathbf{H}$ during training. Bottom: During prediction, the current context $\mathbf{v}_{\text{context}}$ is compared to the normalized memory matrix $\mathbf{H}_{\text{norm}}$ and the best matching state w_i is chosen as a hypothesis for the next sequence state.

sequences will be observed in the data. In this case, d-dimensional distributed vectors can approximate joint distributions of larger state spaces and longer temporal lags than the trivial use of strictly orthogonal codes (m x $A >> d$), making the approach scalable for potentially huge state spaces (e.g., language modeling) using a fixed amount of memory resources. We use the basic idea of RI memory matrix to store the preceding context $w_{t-m},\ldots, w_t$ of each state w_{t+1} and retrieve the most likely next state during prediction by comparing the current context and the previously learned context representations. For this to work, the context vectors $\mathbf{v}_{\text{context}}$ must capture the structural information of the typical preceding states, including the state identities, their temporal ordering, and the relative importance of the states at different temporal distances in the history. Also, the vector representation should be robust so that it can generalize to only partially familiar sequences, and ideally, the quality of the representation should not critically dependent on the length of the temporal history included in the representation (the model order).

The proposed method is also much faster than any SDM-based approach for sequential prediction since the time-consuming distance computations with a huge address matrix are not required. Prediction and training in SDM require $O\{Md\}$ distance computations whereas training in RI does not require computation of distances between hypervectors at all,

and prediction requires only O{Ad} computations with $A << M$ for any typical data set with alphabet size A.

First, a mapping from each sequence state w_{t-k} to a sparse hyperdimensional vector $\mathbf{v}_{t-k}$ is defined so that $\mathbf{v}_{t-k} = f(w_{t-k}, k)$. Unlike the standard RI, a unique sparse vector is randomly generated for each possible sequence state w_i and for each possible lag $k \in \{0, 1, 2, \ldots, m\}$ so that the same state w_i obtains a different hypervector representation at different temporal lags with respect to the current time t (Fig. 1; cf. shift register idea in [2]). For notational simplicity, these lag-specific vectors are referred to as $\mathbf{v}_{t-k}$ (see Fig. 1). Now, a sequence of length L can be uniquely described using a sum vector $\mathbf{v}_{\text{seq}}$:

$$\mathbf{v}_{\text{seq}} = \sum_{k=0}^{L} \mathbf{v}_{L-k} \tag{6}$$

In the context of prediction, it is desirable to represent only a finite preceding context of the sequence up to a history length m. Also, as motivated earlier, the preceding states should receive weights corresponding to their predictive power. This leads to a formulation of the context $\mathbf{v}_{\text{context}}$ at time t:

$$\mathbf{v}_{\text{context}} = \sum_{k=0}^{m-1} \lambda_k \mathbf{v}_{t-k} \tag{7}$$

In order to derive the lag-specific weights λ_k, the mutual information function (MI) [31] is utilized. The average MI at lag k is expressed as

$$\text{MI}_k = \sum_{i,j} P_k(w_i, w_j) \log_2 \frac{P_k(w_i, w_j)}{P(w_i)P(w_j)} \tag{8}$$

In the equation, $P_k(w_i, w_j)$ is the probability of a state pair $\{w_i, w_j\}$ when w_j is delayed by k elements with respect to w_i. What the MI essentially measures is the amount of statistical dependency in the signal at different temporal lags k and accounts also for non-linear dependencies between signal states. The assumption in the current work is that the average statistical dependency at distance k equals to the average predictive power from that distance relative to other possible lags. Since the MI is systematically overestimated from finite length sequences, the following correction is made to the empirically estimated MI' in order obtain the final lag-specific weights [32-34] (see also [35] and references therein):

$$\lambda_{k-1} = \text{MI}'_k - \frac{(A-1)^2}{2L} \tag{9}$$

Note that since the goal is to predict w_{t+1} from w_t, the reliability λ_0 of the state w_t is not MI_0 but the standard bi-gram dependency MI_1. Also note that for any non-deterministic process, the value of MI decays to zero as k increases. This provides a natural upper bound for the model order m used in the prediction since the use of information from longer distances in the history has no effect on the prediction result. In the special case of a first-order Markov process, the MI-based weights decay in a geometric fashion similarly to the parameters in [2] and [3]. For all other temporal dependency structures, the MI provides a more accurate description of the predictive power of the preceding states than the geometric weighting scheme.

B. Predicting from a hypervector memory

Unlike the RI where the goal is to compute the synonymy of two words, the goal here is to predict the most likely next state of a sequence, i.e., to estimate the distribution of the form $P(w_{t+1} \mid w_t, w_{t-1}, \ldots, w_{t-m+1})$. In order to do this, a matrix $\mathbf{H}$ of size M x d is initialized similarly to the RI (Fig. 3). Then the preceding temporal context ("left context") of each observed sequence state w_t is computed from w_{t-m+1} up to w_t using Eq. (7) and summed to the corresponding row of $\mathbf{H}$ defined by w_{t+1} as in the normal RI training. In this manner, the rows of $\mathbf{H}$ become overall compositional descriptions of the sequence histories for the states w_i.

In order to compute the non-normalized probability distribution $\mathbf{p}$ of next states, one simply computes the matrix product of the currently observed context $\mathbf{v}_{\text{context}}$ and the memory matrix $\mathbf{H}$,

$$\mathbf{p} = \mathbf{H}\mathbf{v}_{\text{context}} \tag{10}$$

and chooses the state with the largest value of $\mathbf{p}$ as the hypothesis w_{hypo} for the next state:

$$w_{\text{hypo}} = \arg_i \max\{p(w_i)\} \tag{11}$$

The raw data in $\mathbf{H}$ basically provides a probability estimator for the next state that is dependent on the frequencies of state occurrences (i.e., a sort of pseudo-maximum-likelihood solution). However, normalization of the columns of $\mathbf{H}$ to unit vectors before applying Eq. (10) was found to slightly improve the average prediction accuracy. From now on, this column-wise normalized sparse distributed predictor will be referred to as SDP-c.

Another possibility to normalize the data in $\mathbf{H}$ is to maximize the prediction accuracy of each individual state. This can be beneficial in applications where the successful prediction of rarely occurring events is more important than getting the most frequent states maximally correct (e.g., predicting when rarely used mobile phone user interface (UI) actions are needed if they reside deep in the UI menu hierarchy in comparison to the frequent actions). In order to do this, the rows of $\mathbf{H}$ are normalized into unit vectors, from now on referred to as the SDP-r variant of the algorithm. With this normalization, Eq. (10) essentially becomes comparable to the cross-correlation between the context history of a state in $\mathbf{H}$ and the current sequence context $\mathbf{v}_{\text{context}}$ (whether $\mathbf{v}_{\text{context}}$ is normalized to a unit vector or not does not affect the shape of the predicted state distribution but simply its scale). Naturally, the difference between SDP-r and SDP-c increases as the state frequency distribution becomes more different from a uniform distribution.

III. EXPERIMENTS

A. Material

Prediction of three qualitatively different types of mobile phone use data was studied in the experiments: clustered GPS locations, application usage, and media player records. All data were extracted from the Nokia Lausanne dataset that was collected during 2009 and 2010 from almost two hundred persons using Nokia N95 mobile phones [4]. All data consists of normal everyday use of the phones with the data collection and transfer software running in the background. The average data collection period per test subject was approximately one year. Since all of the data had been collected in advance instead of running a predictor on-line in the phone, there is no interaction between any predictor performance and the phone use patterns.

For all data types, data from users with length less than L = 200 samples or having less than A = 6 unique states were excluded in order to avoid ill-defined or trivial prediction problems. The chronologically first 80% of the available data points were always used for training of the prediction model while the remaining 20% were used to evaluate the prediction accuracy. The prediction models were always for user-specific data and no generalization or model combination across multiple users was attempted.

In the Lausanne dataset, the GPS data has been collected by turning the GPS receiver on and off by using a number of system internal heuristics, leading to asynchronous sampling of the user location [4]. In order to represent phone user location in a discrete sequential form, the raw GPS data (latitude and longitude) were converted into points of interest (POIs). This was done by recursively splitting the two-dimensional data cloud into two halves along the direction of the second principal component (covariance matrix eigenvector with the second largest eigenvalue) so that each half contains equal number of data points. After N splits, 2^N regions were obtained in the feature space. All points of a single region were then used to compute the mean centroid of that region, and all GPS data points were assigned to the nearest centroid in terms of Euclidean distance (see [36]). The result is somewhat similar to the standard k-means clustering, but the process is much faster with a large amount of data and leads to a more uniform distribution of data points across all clusters. Temporally subsequent repetitions of the same POIs caused by a stationary phone user were removed from the sequences since the goal was to predict the most likely next location of the user. After pre-processing, the average number of GPS samples per test subject was 3385 (min 240, max 36521, highly non-Gaussian distribution across different test subjects). GPS data exceeding the minimum of 200 data points were available from a total of 167 users.

As for the application use, applications that were launched more than 20 times during the recording period were included in the application dataset. However, *screensaver* and *standby* applications were excluded from the sequences since they reflect automatic behavior of the phone and had a very high frequency of occurrence compared to user initiated applications. Each application identifier was assigned with a unique integer value, thereby converting the application use logs into discrete temporal sequences from a finite alphabet. The average number of applications per user was 18.0 (standard deviation ± 6.8) and the average number of all app uses was 4017 (± 2951). Typical frequently used applications included, e.g., *text messaging*, *contacts*, *telephone*, *log, camera*, *calendar, clock, web browser, maps, e-mail, gallery, settings,* and *music player*. Application-use data were available from a total of 171 users.

In the prediction of music listening patterns, the original phone logs contain a sequence of songs played by the user during the data collection period. Each song is associated with metadata containing the artist name, album title, and the track title. In the current study, only the artist information was used so that each state of the media player sequence corresponds to a unique artist played with the phone. Moreover, subsequent repetitions of the same artist were removed in order remove the effects of listening to an entire album at once and to simulate the process of providing suggestions of the next artist to the user. The mean number of artists per user was 27.0 (± 16.2) with the mean number of samples being 1039 (min 217, max 7047). Media player data were available from a total of 32 users.

B. Evaluation

The prediction accuracy was measured in terms of unweighted average recall (UAR) and weighted average recall (WAR). UAR is defined as the average of class specific prediction accuracies (the mean of a confusion matrix diagonal) whereas WAR is simply the ratio of correct predictions to the total number of predicted states. UAR provides a more balanced view of the prediction accuracy for highly uneven class distributions and is therefore often more beneficial in the evaluation of pattern recognition algorithms (e.g., [37]). In the prediction of mobile phone application use, UAR better reflects the system's ability to recognize situations where rarely used applications are needed, whereas a relatively high WAR can be achieved by simply predicting the most frequent applications correctly (in principle, if an application such as "browser" is used 60% of the time, a WAR of 60% can be achieved by always guessing <browser> for the next application even if there are tens of other possibilities). On the other hand, WAR is a straightforward measure of how often the prediction is correct in total.

UAR and WAR were estimated separately for each phone user in the data and the averages of the measures were computed across all users. For application and media player use, the evaluation was also performed by studying how often the true next state is within the five best hypotheses provided by the algorithm. This type of accuracy is relevant in applications where the phone user is provided with several suggestions for the next apps or pieces of music.

C. Compared baseline methods

The two SDP variants were compared against the standard bi-, tri-, and four-gram models as n-grams are the most

straightforward and computationally efficient way to learn a predictive model for Markov processes of a given order. N-gram probabilities were estimated from the state frequencies

$$P(w_{t+1} \mid w_t, w_{t-1}, \ldots, w_{t-n+2}) = \frac{C(w_{t+1}, w_t, w_{t-1}, \ldots, w_{t-n+2})}{C(w_t, w_{t-1}, \ldots, w_{t-n+2})} \quad (12)$$

where $C(w_t)$ is the frequency of state w_t occurrence in the training data. Even though the n-grams are simple and suffer from the exponential increase in parameters for higher order modeling, they are still widely used in various predictive applications due to their simplicity and computational speed. Since the idea here is to compare SDP to a fixed-order model, no n-gram smoothing was performed and previously unseen n-grams always led to a random prediction during testing.

In addition, the mixed-order Markov chain model [9] was used as a state-of-the art baseline system for modeling variable-order structure (see section I). Note that the mixed-order model is no longer incremental, but provides a reference performance level of an approach that is known to be capable of capturing varying-order temporal structure. In the current work, the transition probabilities of the model were always initialized from lagged bi-gram probabilities and the mixing weights $\lambda_k(w)$ were initialized as a uniform distribution. The EM algorithm was always run for 6 iterations as it was found to provide a reasonable trade-off between log-likelihood convergence on the training data and generalization of the model to the test data.

A. Results

Results for all data types were computed with SDPs using a vector dimensionality $d = 2000$ with 5% of vector values non-zero (±1). Figs. 4 and 5 show the application prediction performance for 1 and 5 best hypotheses, respectively. Figs. 6 and 7 show GPS POI prediction accuracy for 8 and 16 POIs, respectively. Finally, Figs. 8 and 9 show media player prediction accuracy for 1 and 5 best hypotheses.

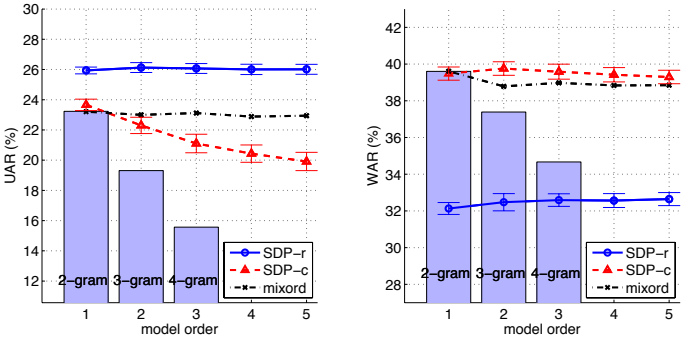

Fig. 4. Application prediction performance as a function of model order (1 best hypothesis). The left panel shows the unweighted average recall (UAR) and the right panel shows the weighted average recall (WAR). The solid line denotes the SDP-r performance whereas the dashed line denotes the SDP-c performance. The dash-dotted line shows the mixed-order Markov model performance and vertical bars show n-gram performance for bi-, tri-, and four-grams. Note that model order m on the x-axis refers to the Markov process order, not the n-gram order that is $n = m+1$. Standard deviations of the SDP variants across multiple trials with different random hypervector assignments are shown with horizontal bars.

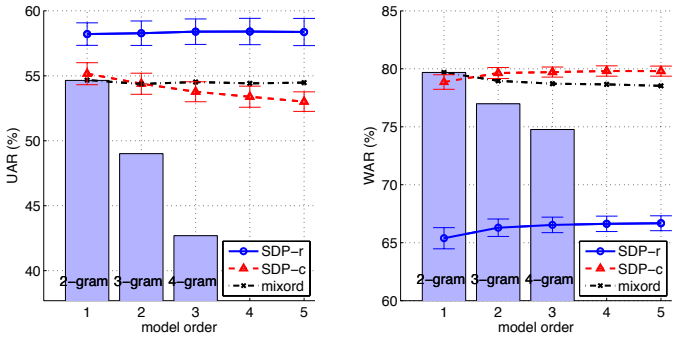

Fig. 5. Application prediction performance (5 best hypotheses). The left panel shows the UAR and the right panel shows the WAR. The solid line denotes the SDP-r performance whereas the dashed line denotes the SDP-c performance. The dash-dotted line shows the mixed-order Markov model performance and vertical bars show n-gram performance for bi-, tri-, and four-grams.

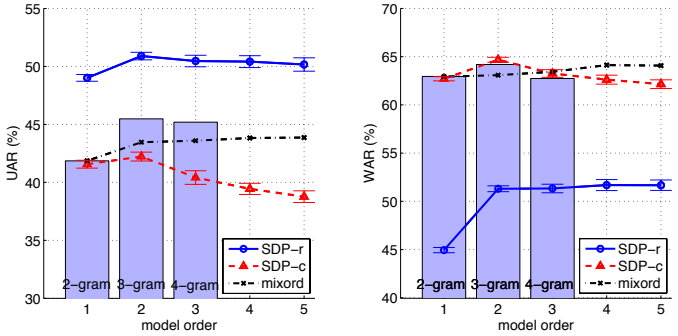

Fig. 6. GPS prediction performance for 8 POIs. The left panel shows the UAR and the right panel shows the WAR. The solid line denotes the SDP-r performance whereas the dashed line denotes the SDP-c performance. The dash-dotted line shows the mixed-order Markov model performance and vertical bars show n-gram performance for bi-, tri-, and four-grams.

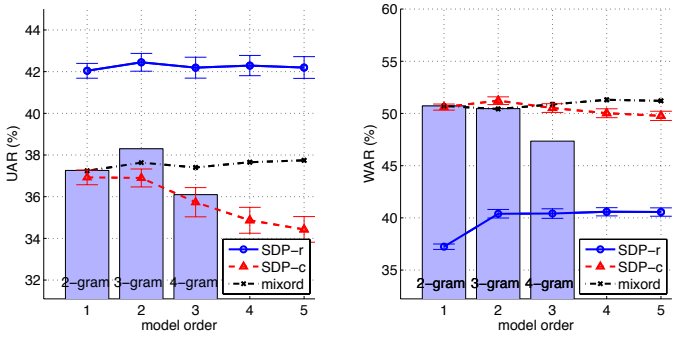

Fig. 7. GPS prediction performance for 16 POIs. The left panel shows the UAR and the right panel shows the WAR. The solid line denotes the SDP-r performance whereas the dashed line denotes the SDP-c performance. The dash-dotted line shows the mixed-order Markov model performance and vertical bars show n-gram performance for bi-, tri-, and four-grams.

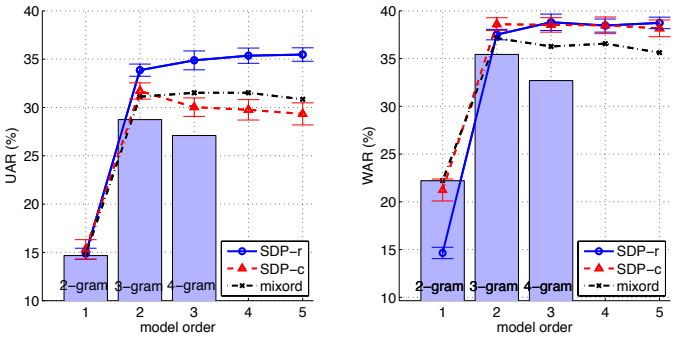

Fig. 8. Media player prediction performance (1 best hypothesis). The left panel shows the UAR and the right panel shows the WAR. The solid line denotes the SDP-r performance whereas the dashed line denotes the SDP-c performance. The dash-dotted line shows the mixed-order Markov model performance and vertical bars show n-gram performance for bi-, tri-, and four-grams.

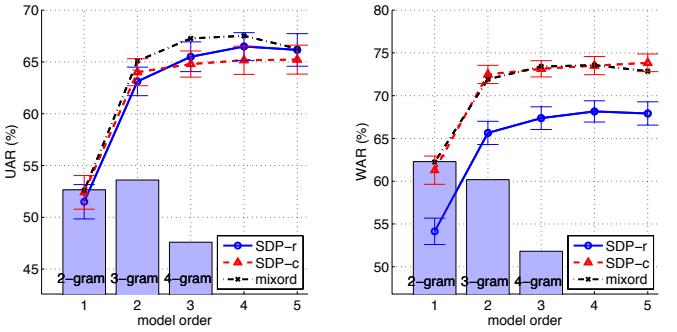

Fig. 9. Media player prediction performance (5 best hypotheses). The left panel shows the UAR and the right panel shows the WAR. The solid line denotes the SDP-r performance whereas the dashed line denotes the SDP-c performance. The dash-dotted line shows the mixed-order Markov model performance and vertical bars show n-gram performance for bi-, tri-, and four-grams.

In the case of SDP, the mean and standard deviation of the performance was measured across 5 runs of the algorithm in order to see how much variation is introduced by the random generation of the state and lag specific hypervectors. The standard deviations are denoted with horizontal bars in each figure. N-grams and the mixed-order Markov model are deterministic for a given set of data and therefore no standard deviations are reported for them.

As can be observed from the results, the two variants of the SDP-predictor perform notably differently in terms of UAR and WAR. SDP-r clearly outperforms all other methods in terms of UAR. The only exception is the case of predicting the five best hypotheses for media player usage where SPD-r achieves a similar level of performance with the SDP-c and the mixed-order Markov model. On the other hand, SDP-c and the mixed-order Markov model perform well in terms of WAR where SDP-r performs poorly except for the media player data. This is an expected result due to the absence of overall state-frequency information in SDP-r, comparable to the absence of a prior in a Bayesian naïve predictor. In terms of WAR, bi- and tri-grams provide solid baseline performance levels for application and GPS prediction, and they are not greatly exceeded by the variable order models. The situation is different in the media player data where the n-grams fall far behind the other methods. Notably, a model order of as high as $m = 4$ seems to provide optimal results in the media player data set, demonstrating the power of variable-order structure modeling for complex asynchronously sampled data.

Fig. 10 shows a comparison of MI-based weighting (current work) and the geometrically diminishing weights proposed by Bose et al. [2] and Snaider & Franklin [3] in the media player prediction task. The geometric weights $\lambda_k = \lambda_0^k$ were optimized to minimize their distance to the MI curve so that, in the case of a first-order Markov process, the fit would be perfect between the MI and the geometric decay. As can be seen from the results, the geometric weighting scheme is unable to capture all the relevant information from higher lags, performing significantly worse than the MI variant. Importantly, the SDP variants are relatively stable with increasing model order even beyond the optimal order in terms of performance. This suggests that the MI-based weighting of information from different lags is successful in utilizing long-distance dependencies, and a precisely optimized model order is not critical for reasonable performance. As an exception to this, the UAR performance of the SDP-c variant seems sometimes to drop notably above the optimal order (Figs. 4–9). The reason for this is currently unknown but suggests that the weighting scheme could be still improved. For example, if the SDP were modified to use an iterative training scheme, the weights λ_k initialized by the MI could be optimized for prediction performance on the training data set using some optimization procedure. However, it is evident from the results that the SDP-r variant should be preferred if high UAR is desired. Also, a combination of SDP-r and SDP-c could be utilized. As for the mixed-order Markov model, the convergence to the optimal performance level is guaranteed by the EM algorithm even for "too high" model orders as long as there is a sufficient amount of training data available.

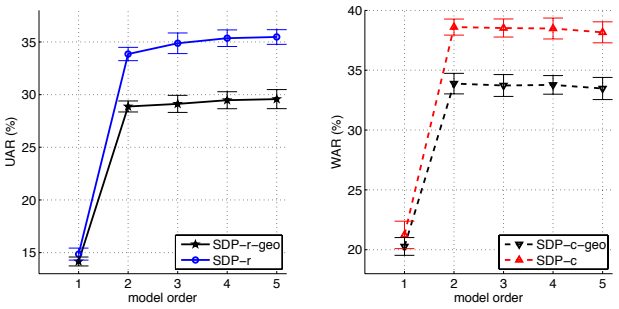

Fig. 10. Comparison of media player prediction performance for MI-based history weights and the geometrically diminishing weights according to [2,3], denoted with SDP-r/c-geo. The left panel shows the UAR while the right panel shows the WAR.

The effect of hypervector dimensionality and sparsity was also studied. Fig. 11 shows the SDP-r prediction performance on the media player dataset as a function of vector dimensionality d with 5% of non-zero elements in each vector. As can be seen from the figure, the performance monotonically increases and finally saturates with an increasing dimensionality. This confirms that the desired properties of the hyperdimensional spaces become apparent when the dimensionality starts to approach thousands (cf. [1]), whereas the sum-code representations fail to maintain details

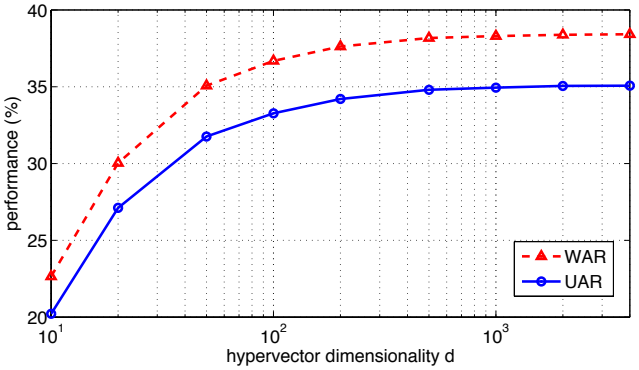

Fig. 11. Media player prediction performance as a function of hypervector dimensionality d with SDC-r and model order $m = 5$. The solid line denotes the UAR (%) and the dashed line denotes the WAR (%).

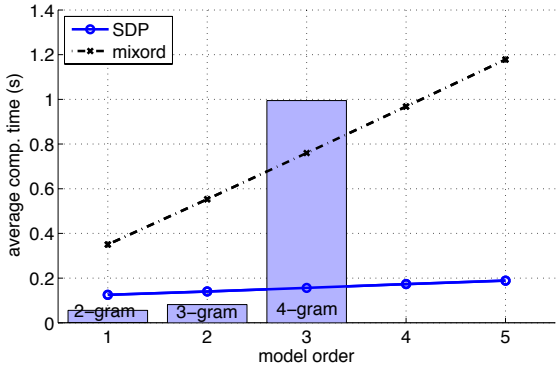

Fig. 12. Average computation times of the compared algorithms as a function of model order (average time in seconds spent for training and recognition of media player data from a single mobile phone user). The algorithms were run in the MATLAB R2012b environment using 4x3.2 GHz Intel Xeon processors using fast but not explicitly parallelized code. The results should not be taken as the final word for computational costs of different algorithms but simply shows the scale of SDP speed in comparison to the other approaches.

of their components with low-dimensional random mappings. On the other hand, the vector sparsity (the number of non-zero elements in a vector) did not affect the SDP performance as long as the number of non-zero elements was between 5% and 100%. As the proportion of non-zero elements approaches zero, the chance risk of creating similar hypervectors for different states increases. However, this problem is not very pronounced with the relatively small number of unique sequence states analyzed in the current study.

Fig. 12 shows the computation times of the compared methods on the media player dataset. The mixed-order Markov model and SDP both have complexity of O{bm} as the number of parameters increases only linearly as a function of model order. However, at least in a naïve implementation of the algorithm, the constant b of the mixed-order model is much larger due to the iterative batch processing of the entire training data while the SDP only processes each data point once. In Fig. 12, the computational complexity of the n-grams increases significantly due to the exponentially increasing number of parameters with an increasing model order. However, high-order n-gram computational costs can also be alleviated with properly optimized data structures for n-gram representation. Importantly, the overall processing time for the SDP is realistic even for a mobile phone platform and can be further facilitated by specialized software or hardware solutions for parallelizing the hyperdimensional memory and operations and by making use of the sparsity of the matrices (e.g., [38,39]).

In addition to the mobile phone data experiments, SDP was also tested in two long time lag prediction problems described in the context of LSTMs [5].

In the first experiment, namely the task of two widely separated symbols (see [5]), the goal of the algorithm is to classify sequences into four classes. All sequences start with symbol B and end to symbol E, otherwise consisting of randomly sampled symbols {a,b,c,d} except for two special symbols at positions t_1 and t_2 that are randomly set to either X or Y. All sequences are 100-110 elements long and t_1 is always randomly chosen between 10 and 20 while t_2 is between 50 and 60. Depending on the identity of the symbols at t_1 and t_2, each sequence belongs to one of the four possible classes (XX → c_1, XY → c_2, YY → c_3, YX → c_4). The goal of the algorithm is to recognize the correct class of each test sequence, given a set of previously observed training sequences. When SDP with maximum lag of m = 50 and dimension d = 2000 was applied to this task, it achieved a correct classification rate of 75% after 30000 training samples (25% chance level) and then saturated in performance. In comparison, a nearly perfect classification rate is achieved with LSTMs after 30000 sequences, but with a manually tailored network architecture for the given problem.

In the second long lag experiment, the goal was to learn embedded Reber grammars [40], that is, finite state automata that subsume the automata itself in a recursive manner. While SDP successfully learns standard (non-embedded) Reber grammars, it fails in the embedded grammar task unlike LSTM that can solve the problem. This failure is likely caused by the highly non-linear dependencies in the data, practically meaning that individual symbols can have deterministic effects on the sequence after several and varying number of intermediate stochastic steps. As SDP cannot differentiate between variability in symbol identity at a specific lag from variability in lags at which a specific symbol occurs, it cannot learn perfect predictive models for clearly non-Markovian processes such as the embedded Reber grammar. Note that also standard (non-LSTM) recurrent networks and n-grams fail in this task (see, e.g., [5]).

IV. Discussion and Conclusions

A new method to predict sequential data using hyperdimensional coding is proposed in this work. The approach can be easily applied to different prediction tasks as it is very robust with respect to its hyperparameters. The MI-based temporal weights are automatically derived from the learning data, leaving the dimension of the hyperspace as the only free hyperparameter. Even for the dimensionality, only graceful degradation will be observed from the ideal model performance when the dimensionality is decreased below the minimum optimal level.

The method has two different ways of normalizing the memory used for prediction, leading to optimization of either

unweighted or weighted average recall. The experiments show that the incremental algorithm is able to utilize high-order temporal structure when it exists, and thereby achieves a performance level that compares well against the iterative mixed-order Markov model.

Due to its incrementality and tolerable computational complexity, the SDP seems a promising choice for real-time applications where storage or transmission of the entire data history is expensive, ruling out methods based on iterative batch training such as the majority of the recurrent neural network architectures. Since both SDP-r and SDP-c variants are applicable using the same previously learned statistics of the sequential data (the non-normalized **H** matrix), the predictor can provide hypotheses for the next state that either maximize UAR or WAR on demand. This type of versatility can be beneficial, e.g., in recommendation engines in mobile phones where the user can be provided with a number of choices that he or she would like to perform next with the phone. In these cases, the most frequent operations should be readily available (cf. WAR), but the system should also be sensitive to situations where rarely used operations are required as they might be located very deep in the UI hierarchy (cf. UAR). By showing a number of SDP-r- and SDP-c-based recommendations, both of these goals can be achieved simultaneously. Usage of the SDP in this type of application is also one of the topics of future studies.

Although the current experiments were limited to the prediction of mobile phone user patterns using finite state spaces, the coding capacity of the hyperdimensional spaces should make SDP also beneficial in other applications such as language models of automatic speech recognition (see [41] for capacity analysis). The study shows that the hyperdimensional coding can be used to represent complex and variable distance temporal dependencies in an efficient and mathematically compact manner. In principle, the same framework should scale to the use of other contextual information sources that are additively coded to the hypervectors representing the contextual states. For example, the app prediction of the current work can benefit from other information, such as, the time of day or location of the user (see, [42]), although these aspects were intentionally left out from the current study for the simplicity of presentation. Also, the current method of weighting different information sources according to their statistical dependency can be combined with the other methods used with hyperdimensional computing.

The major limitation in SDP is that it still only learns an approximation of Markov-processes up to some finite order using episodic descriptions of the sequences. This means that the algorithm cannot capture complex long-distance regularities such as those present in embedded grammars (see [5,40]) that can be successfully solved using LSTMs. Since weighting of past information in prediction is based on the average temporal structure (MI) of the data, the model is unable to take into account non-Markovian characteristics such as grammar-like recursions that might be responsible for generating the data. On the other hand, the existing state-of-the-art approaches also require careful model architecture selection and initial parametrization in order to successfully solve these tasks (see [5]), making their applicability to different prediction tasks non-trivial for naïve users or without any a priori knowledge of the data properties.

Interestingly, Plate [43,44] has shown how nested hierarchical compositional structures can be represented within the hypervectors by using circular convolution, and how these representations are similar to the analogical reasoning in humans. In [45], it is shown how distributed representations can be used to enhance the performance of information retrieval in knowledge-based systems. These links provide interesting possibilities for extending the simple linear state history representation into more comprehensive contextual state representations that utilize hierarchical complex information from a number of different information sources. This may also help the approach to overcome the current limitations in learning more complex grammar-like properties of the input data.

Finally, the current work has only concentrated on the modeling of discrete states with their sparse codes being pseudo-orthogonal with each other. With a proper mapping, the hyperdimensional coding also enables similarity-based computations where the hypervectors that have a similar meaning or similar value in the original signal space (e.g., neighboring GPS coordinates) can be represented using partially overlapping sparse codes. This would allow the utilization of the second primary advantage of hyperdimensional spaces, namely the content addressability, in multivariate feature spaces in addition to the presently investigated discrete domain.

ACKNOWLEDGMENT

The authors would like to thank Leo Kärkkäinen from Nokia Research Center for useful comments and ideas related to this study and Olivier Bornet from the Idiap / EPFL for his help with the Lausanne database.

REFERENCES

[1] P. Kanerva, "Hyperdimensional Computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[2] J. Bose, S. B. Furber, and J. L. Shapiro, "An associative memory for the on-line recognition and prediction of temporal sequences," in *Proc. IEEE International Joint Conference on Neural Networks*, Montreal, Canada, Jul./Aug. 2005, pp. 1223–1228.

[3] J. Snaider and S. Franklin, "Extended Sparse Distributed Memory and Sequence Storage," *Cognitive Computation*, vol. 4, no. 2, pp. 172–180, 2012.

[4] N. Kiukkonen, O. Dousse, D. Gatica-Perez, and J. Laurila, "Towards rich mobile phone datasets: Lausanne data collection campaign," in *Proc. ACM International Conference on Pervasive Services (ICPS'2010)*, Berlin, Germany, Jul. 2010.

[5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] A. Graves, A. Mohamed, and J. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP'13*, Vancouver, Canada, May 2013, pp. 6645–6649.

[7] F. Jelinek, "Continuous Speech Recognition by Statistical Methods", *Proceedings of the IEEE*, vol. 64, pp. 532–556, Apr. 1976.

[8] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 35, pp. 400–401, Mar. 1987.

[9] L. Saul and F. Pereira, "Aggregate and mixed-order Markov models for statistical language processing," in *Proc. Second Conference on Empirical Methods in Natural Language Processing*, Providence, RI, Aug. 1997, pp. 81–89.
[10] A. E. Raftery, "A Model for High-order Markov Chains," *J. Royal Statistical Society. Series B (Methodological)*, vol. B47, no. 3, pp. 528–539, 1985.
[11] A. Berchtold and A. E. Raftery, "The Mixture Transition Distribution Model for High-Order Markov Chains and Non-Gaussian Time Series," *Statistical Science*, vol. 17, no. 3, pp. 328–356, 2002.
[12] A. Prinzie and D. Van den Poel, "Investigating purchasing-sequence patterns for financial services using Markov, MTD and MTDg models," *European Journal of Operational Research*, vol. 170, no. 3, pp. 710–734, 2006.
[13] M. J. Weinberger, J. Rissanen, and M. Feder, "A universal finite memory source," *IEEE Trans. Inform. Theory*, vol. 41, pp. 643–652, May 1995.
[14] P. Bühlmann and A. J. Wyner, "Variable length Markov Chains," *Ann. Statistics*, vol. 27, no. 2, pp. 480–513, 1999.
[15] P. Kanerva, J. Kristoferson, and A. Holst, "Random indexing of text samples for latent semantic analysis," in *Proc. 22nd Annual Conference of the Cognitive Science Society,* Philadelphia, PA, Aug. 2000, pp. 1036.
[16] M. Sahlgren, "An introduction to random indexing", in *Proc. Methods and Applications of Semantic Indexing Workshop, 7th Int. Conf. on Terminology and Knowledge Engineering,* Copenhagen, Denmark, Aug. 2005.
[17] P. Kanerva, *Sparse distributed memory*. Cambridge, MA: Bradford/MIT Press, 1988.
[18] Y.-S. Hong and S.-S. Chen, "Character recognition in a sparse distributed memory," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 21, pp. 674–678, May/Jun. 1991.
[19] U. Ramamurthy, S. K. D'Mello, and S. Franklin, "Modified sparse distributed memory as transient episodic memory for cognitive software agents," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics,* The Hague, Netherlands, Oct. 2004, pp. 5858–5863.
[20] R. W. Prager and F. Fallside, "The modified Kanerva model for automatic speech recognition," *Computer Speech & Language*, vol. 3, no. 1, pp. 61–81, 1989.
[21] D. G. Danforth, *An empirical investigation of sparse distributed memory using discrete speech recognition*. RIACS Technical Report 90.18, NASA Ames Research Center, 1990.
[22] S. Jockel, M. Mendes, J. Zhang, P. Coimbra, and M. Crisóstomo, "Robot navigation and manipulation based on a predictive associative memory," in *Proc. 8th IEEE International Conference on Development and Learning* (*ICDL'09*), Shanghai, China, Jun. 2009, pp. 1–7.
[23] H. Meng, K. Appiah, A. Hunter, S. Yue, M. Hobden, N. Priestley, P. Hobden, and C. Petitt, "A modified sparse distributed memory model for extracting clean patterns from noisy inputs," in *Proc. Int. Joint Conference on Neural Networks (IJCNN'09)*, Atlanta, GA, Jun. 2009, pp. 2084–2089.
[24] P. Kanerva, "Sparse distributed memory and related models," In *Associative Neural Memories: Theory and Implementation*, M. Hassoun, Ed., New York: Oxford University Press, 1993, pp. 50–76.
[25] J. T. Abbott, J. B. Hamrick, and T. L. Griffiths, "Approximating Bayesian inference with a sparse distributed memory system," in *Proc. Annual Conference of the Cognitive Science Society*, Berlin, Germany, Jul./Aug. 2013, pp. 1686–1691.
[26] S. Jockel, *Crossmodal learning and prediction of autobiographical episodic experiences using a sparse distributed memory*. Doctoral Thesis, University of Hamburg, Department of Informatics, 2010.
[27] B. Ratitch and D. Precup, "Sparse distributed memories for on-line value-based reinforcement learning," *Lecture Notes in Computer Science,* vol. 3201, no. 1, pp. 347–358, 2004.
[28] T. A. Hely, D. J. Willshaw, and G. M. Hayes, "A new approach to Kanerva's sparse distributed memory," *IEEE Trans. Neural Networks*, vol. 8, pp. 101–105, May 1997.
[29] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge," *Psychological Review*, vol. 104, no. 2, pp. 211–240, 1997.
[30] M. Terho, "Practical approach to real time contextual data access," In *Frontiers in artificial intelligence and applications, volume 251: Information modelling and knowledge bases XXIV*, P. Vojtás et al., Eds. IOS Press, Netherlands, pp. 328–343, 2013.
[31] W. Li, "Mutual information functions versus correlation functions," *J. Statistical Physics*, vol. 60, no. 5/6, pp. 823–837, 1990.
[32] H. Herzel, A. O. Schmitt, and W. Ebeling, "Finite sample effects in sequence analysis," *Chaos, Solitons and Fractals*, vol. 4, no. 1, pp. 97–113, 1994.
[33] H. Herzel and I. Grosse, "Measuring correlations in symbol sequences," *Physica A*, vol. 216, no. 4, pp. 518–542, Jul. 1995.
[34] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig, "The mutual information: Detecting and evaluation dependencies between variables," *Bioinformatics*, vol. 5 suppl. 2, pp. S231–240, 2002.
[35] C. O. Daub, R. Steuer, J. Selbig, and S. Kloska, "Estimating mutual information using B-spline functions – an improved measure for analyzing gene expression data," *BMC Bioinformatics*, vol. 5, no. 118, 2004.
[36] Y.-W. Chen and C.-C. Chen, "Vector quantization by principal component analysis," in *Proc. Vision Interfaces*, Vancouver, Canada, Jun. 1998, pp. 295–299.
[37] B. Schuller, S. Steidl, A. Batliner, E. Nöth, A. Vinciarelli, F. Burkhardt, R. van Son, F. Weninger, F. Eyben, T. Bocklet, G. Mohammadi, and B. Weiss, "The INTERSPEECH 2012 speaker trait challenge," in *Proc. Interspeech'12*, Portland, OR, Sept. 2012.
[38] T. Hämäläinen, P. Kolinummi, and K. Kaski, "Linearly expandable partial tree shape architecture for parallel neurocomputer," in *Proc. International Conference on Artificial Neural Networks (ICANN'96)*, Bochum, Germany, Jul. 1996, pp. 365–370.
[39] L. Kärkkäinen, M. Terho, and N. Werdi, "Method and apparatus for providing efficient context classification," US Patent application 20120110267 A1, May 5, 2012.
[40] A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland, "Finite state automata and Simple Recurrent Networks," *Neural Computation*, vol. 1, no. 3, pp. 372–381, 1989.
[41] S. I. Gallant and T. W. Okaywe, "Representing Objects, Relations, and Sequences," *Neural Computation,* vol. 25, no. 8, pp. 2038–2078, 2013.
[42] C. Shin, J.-H. Hong, and A. K. Dey, "Understanding and prediction of mobile application usage for smart phones," in *Proc. UbiComp'12*, Pittsburgh, PA, Sep. 2012, pp. 173–182.
[43] T. Plate, "Holographic reduced representations," *IEEE Trans. Neural Networks,* vol. 6, pp. 623–641, May 1995.
[44] T. Plate, "Analogy retrieval and processing with distributed vector representations," *Expert Systems: The International Journal of Knowledge Engineering and Neural Networks – Special Issue on Connectionist Symbol Processing*, vol. 17, no. 1, pp. 29–40, 2000.
[45] D. A. Rachkovskij and S. V. Slipchenko, "Similarity-based retrieval with structure-sensitive sparse binary distributed representations," *Computational Intelligence*, vol. 28, no. 1, pp. 106–129, 2012.

Okko J. Räsänen was born in Finland in 1984. He received the M.Sc. degree in language technology from the Helsinki University of Technology in 2007 and D.Sc. (Tech.) degree in language technology from Aalto University, Finland, in 2013.

He is currently a postdoctoral researcher at the Department of Signal Processing and Acoustics at Aalto University and a visiting researcher at the Language and Cognition Lab of Stanford University. His research interests include computational modeling of language acquisition, cognitive aspects of language processing, context-aware computing, multimodal data analysis, and speech technology in general. He is a member of ISCA and Cognitive Science Society.

Jukka P. Saarinen was born in Finland in 1961. He studied computer architecture, signal processing, telecommunications, and software engineering at Tampere University of Technology, Finland, where he received M.Sc. degree (with honors) in 1986, a Licentiate degree in Technology in 1989, and a Doctor of Technology degree in 1991.

He held the position of a professor of computer engineering at the Tampere University of Technology, where he was the head of the laboratory in 1996-2001. In 2001, he started as the head of Speech and Audio Systems Laboratory in Nokia Research Center. After that he was heading the Audio-Visual Technology Laboratory in 2002-2003 and Multimedia Technologies Laboratory in 2004-2006. Since then, he has been in different positions at Nokia including Nokia Research Fellow and Director of Open Innovation. His research interests cover different signal processing algorithms and applications, multimedia architectures & systems, and DSP architectures. He has published more than 57 international refereed journal articles, 201 refereed international conference papers, and 38 other technical reports. He has also supervised 12 doctoral theses and more than 150 M.Sc. theses.