Learning Correlation Functions on Mixed Data Sequences for Computer Architecture Applications

Zerong Xi, Gita Sukthankar
Department of Computer Science
University of Central Florida
Orlando, FL USA
zxi@knights.ucf.edu, gitars@eecs.ucf.edu

ABSTRACT

Computer architectures require prediction to manage memory and storage operations such as prefetching, caching, and scheduling. Rather than relying on heuristics, a more powerful approach is to employ machine learning to learn general models for storage and retrieval. This paper introduces a technique, Temporally Aware Embedding (TAE), for learning correlation functions directly from mixed data sequences. We use a force-based learning model in which the co-occurrence of data elements within a sliding temporal window creates attraction forces that increase the correlation magnitude, while repulsion forces act to distribute points more uniformly across the embedding space. Our experiments show that TAE outperforms both simple and state of the art strategies at predicting the next element of data access traces. The embeddings learned with our approach can be combined with other algorithms to perform prefetching and caching in architectures which require greater intelligence to keep pace with growing user storage demands.

KEYWORDS

correlation functions; ML for computer architecture

1 INTRODUCTION

Modern computer systems need to rapidly process large amounts of streamed and stored data. To reduce latency, most architectures make predictions about future data demands based on observed sequences in order to make prefetching, caching, and storage decisions [5]. Although there have been algorithms [4, 6] and heuristics developed to handle specific tasks, all of them share the same mathematical characteristic—the need to compute correlation functions from mixed data sequences. A correlation function provides a measure of statistical correlation between random variables, based on spatial or temporal distance [3]. Our aim is to rapidly learn a correlation function from limited data to model the temporal patterns that occur in data streams when related data is accessed.

This paper introduces a method, Temporally Aware Embedding (TAE), for learning an embedding that leverages the temporal information extracted from data retrieval sequences. Since our algorithm solely relies on sampled sequences, it can be used even in complex architectures where the data semantics are not known to the observer. Our assumptions are: 1) several retrieval processes occur concurrently resulting in a single mixed sequence and 2) the correlation is roughly proportional to the first order Markov statistics of the random process.

TAE's correlation embedding is calculated by applying a kernel function to a sliding window over the temporal sequence. Data co-occurrences within the sliding window result in attractive forces that increase the magnitude of correlation while repulsive forces operate to distribute the data more uniformly. Our method yields an informed embedding which can be utilized in combination with other heuristics and machine learning techniques in order to intelligently prefetch, cache, or store data. Our experiments demonstrate that it outperforms both simple heuristics and state of the art deep learning strategies at predicting the next element of real database sequences as well as synthetically generated datasets.

The next section describes related work in the area of sequence analysis for computer architectures. Then we introduce our method, Temporally Aware Embedding (TAE), and present the results of our comparison. We conclude the paper by discussing future extensions of our method to cases where the training dataset has missing elements.

2 RELATED WORK

Sequential data is commonly analyzed using two different types of approaches: pattern mining [2] and sequence learning [8]. Sequence pattern mining extracts frequently repeated subsequences that exceed a minimum support threshold in the dataset. This style of analysis has been employed to find block correlations in storage systems [6]; the C-Miner system extracts association rules based on subsequences found using Closed Sequential Pattern Mining. These rules were then integrated into correlation based prefetching and disk layout systems.

The prefetching problem is particularly well suited for sequence learning since the aim is simply to predict the next element that needs to be retrieved. This can either be structured as a classification or a regression problem and handled with the same type of neural network architectures that are commonly used in natural language processing [7]. Hashemi et al. ([2018]) demonstrated the application of two LSTM models for prefetching: a single embedded LSTM (the benchmark in our paper) and a multi-task LSTM learned from clustered data. The key advantage of our method vs. the above approaches is that it both requires less data to train and less compute time to execute, making it highly suitable for integration into a computer architecture.

3 METHOD

This section provides a description of our algorithm, Temporally Aware Embedding (TAE), for learning embeddings from mixed sequential data.

3.1 Problem Statement

Assume that there is a set $\mathbb{S}$ such that every $a, b \in \mathbb{S}$ has an underlying correlation $cor(a, b) \in \mathbb{R}$ and that $\mathcal{D} = (x_1, ..., x_T) \in \mathbb{S}^{\mathbb{N}}$ is a sequence sampled on $\mathbb{S}$ subject to $P(x_{t+1}|x_t) \propto \frac{cor(x_t, x_{t+1})}{\sum_{x \in \mathbb{S}} cor(x_t, x)}$. TAE embeds every $a \in \mathbb{S}$ based solely on whether a appears in $\mathcal{D}$ into a vector space $\mathbb{V}$ in which $d(v_a, v_b)$ is negatively correlated to $cor(a, b)$. Our desiderata are to create an algorithm that is 1) highly robust to noise and 2) able to deal with a random mixture of multiple sequences $\mathcal{D}_{mix} \in \mathbb{S}^{\mathbb{N}}$.

3.2 Learning the Embedding

First we sample one point $v_x \in \mathbb{V}$ for each $x \in \mathbb{S}'$ from an initial distribution (Gaussian or uniform) where $\mathbb{S}' = \{a \in \mathbb{S} | a \in \mathcal{D}\}$. Then we apply a physics based model of attraction and repulsion forces to modify the position of the points based on their co-occurrence in the observed data sequence. The final embedding is obtained after the model converges to a stable state.

The only information source that we use to aggregate correlated points is the data sequence, $\mathcal{D}$. Intuitively, every pair of consecutive points $v_{x_t}, v_{x_{t+1}}$ should be considered for aggregation. However, due to the existence of noise and interleaved sequences, consecutive points may be separated by irrelevant ones. Instead, we assume that two points $v_{x_t}, v_{x_{t+i}}$ are correlated if $i \leq l$ for some preselected $l \in \mathbb{N}$. Even with our mixed data sequences, if two points appear together within a window of length l in $\mathcal{D}$ multiple times they are likely to be the result of the same data access process.

Specifically, at each $t = 1, ..., T$, we take a weighted sum of the points $v_{x_{t-l}}, ..., v_{x_{t-1}}$, based on their temporal distances to v_{x_t} in $\mathcal{D}$, as the source of an attractive force. Then v_{x_t} moves a distance in $\mathbb{V}$ scaled by the factor $\alpha \in \mathbb{R}$, which may decay over time, as the result of applying the force. A kernel $\mathcal{K} = (k_1, ..., k_l)$, which is either linear or exponential in our experimental settings, serves as the weighting function. Separate kernels can be applied to different dimensions of $\mathbb{V}$ to explore multiple types of correlations. The attraction formula is shown in Equation 1.

$$v_{x_t} := (1 - \alpha) v_{x_t} + \alpha \sum_{i=1:l} k_i v_{x_{t-i}} \tag{1}$$

A space created by attractive forces would collapse resulting in lost volume as well as damaging the smoothness of functions. To prevent this, we incorporate repulsive forces to distribute points more uniformly. A simple idea is to linearly expand $\mathbb{V}$ to maintain a standard such as normalizing each dimension by the standard deviation. This can work well but often embeds most points into a smaller area, while drifting a minority too far. Thus we introduce a repulsive force which is inversely proportional to a power γ of distance and scaled by a factor $\beta \in \mathbb{R}$, as shown in Equation 2. In practice, we only consider the repulsion force between neighboring points to reduce the computation time.

$$v_{x_t} := v_{x_t} + \beta \sum_{x \in \mathbb{S}'} (v_{x_t} - v_x) \cdot d^{-\gamma}(v_{x_t}, v_x) \tag{2}$$

Finally, we obtain an embedding table in which every $a \in \mathbb{S}'$ has a corresponding point $v_a \in \mathbb{V}$. Note the embedding will not encompass elements that were not observed during the data access sequence.

Algorithm 1 Temporally Aware Embedding

Given mixed sequence $\mathcal{D}$, the desired number of mapping dimensions n, kernel $\mathcal{K}$, attraction factor α, repulsion factor β, repulsion power γ, stopping criterion τ
$\mathbb{S} \leftarrow$ UniqueElements($\mathcal{D}$)
for $s \leftarrow \mathbb{S}$ **do** $\mathcal{M}(s) \leftarrow \mathcal{N}(d; \mathbf{0}, \mathbf{I})$
end for
while true do
  $\mathcal{M}' \leftarrow \mathcal{M}$
  $\mathcal{M}'' \leftarrow$ Attractive($\mathcal{D}, \mathbb{S}, \mathcal{K}, \mathcal{M}', n$)
  for $s \leftarrow \mathbb{S}$ **do** $\mathcal{M}(s) \leftarrow (1 - \alpha) \cdot \mathcal{M}'(s) + \alpha \cdot \mathcal{M}''(s)$
  end for
  $\mathcal{M}'' \leftarrow$ Repulsive($\mathcal{M}', \mathbb{S}, \gamma$)
  for $s \leftarrow \mathbb{S}$ **do** $\mathcal{M}(s) \leftarrow \mathcal{M}(s) + \beta \cdot \mathcal{M}''(s)$
  end for
  if $\sum_{s \in \mathbb{S}} d(\mathcal{M}(s), \mathcal{M}'(s)) < \tau$ **then break**
  end if
end while
return $\mathcal{M}$

Algorithm 2 Attraction Force

function Attractive($\mathcal{D}, \mathbb{S}, \mathcal{K}, \mathcal{M}, n$)
  $l \leftarrow$ Length($\mathcal{K}$)
  $c \leftarrow 0^{\mathbb{S}}$
  $\mathcal{M}' \leftarrow 0^{\mathbb{S} \times n}$
  for $i \leftarrow l...$ **do**
    $\mathcal{M}'(\mathcal{D}_i) \leftarrow \mathcal{M}'(\mathcal{D}_i) + \sum_{j=i-l:i} \mathcal{K}_j \mathcal{M}(\mathcal{D}_{i-j})$
    $c_s \leftarrow c_s + 1$
  end for
  for $s \leftarrow \mathbb{S}$ **do** $\mathcal{M}'(s) \leftarrow \mathcal{M}'(s)/c_s$
  end for
  return $\mathcal{M}'$
end function

Algorithm 3 Repulsion Force

function Repulsive($\mathcal{M}, \mathbb{S}, \gamma$)
  $\mathcal{M}' \leftarrow 0^{\mathbb{S} \times n}$
  for $s \leftarrow \mathbb{S}$ **do**
    $\mathcal{M}'(s) \leftarrow \sum_{s' \in \mathbb{S}} (\mathcal{M}(s') - \mathcal{M}(s)) \cdot d^{-\gamma}(\mathcal{M}(s), \mathcal{M}(s'))$
  end for
  return $\mathcal{M}'$
end function

3.3 Correlation-based Prediction

We believe that Temporally Aware Embedding produces correlation functions that can be used by a variety of data access operations in combination with other algorithms. For prefetching, an obvious implementation is to retrieve the N most correlated elements given $u(a)$ where $a \in \mathbb{S}$. A naive approach is to calculate $d(v_a, v_b)$ for every $b \in \mathbb{S}'$ based on the embedding table that we obtain from TAE and prefetch the element with the least distance to v_a.

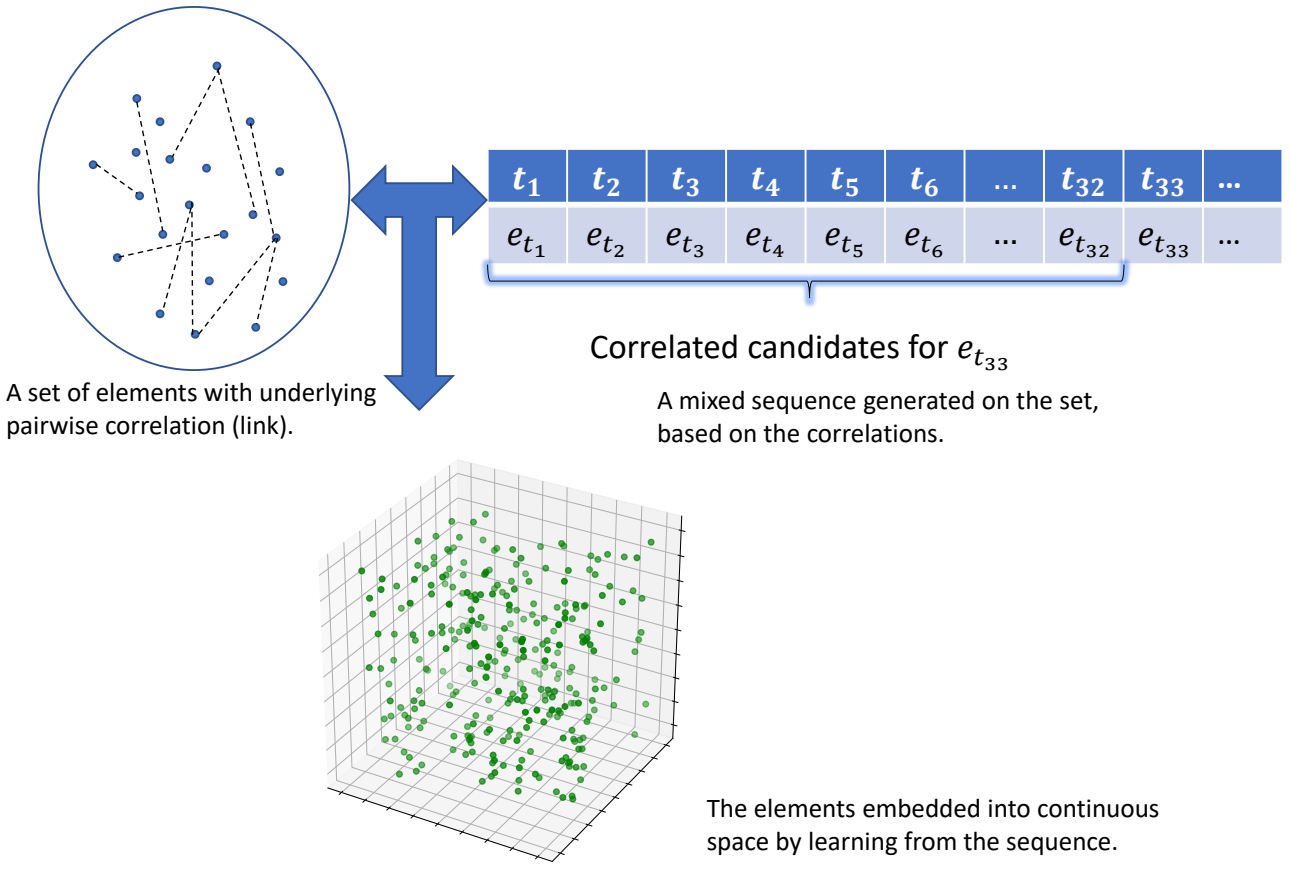

Figure 1: The mixed data sequence (upper right) is generated by multiple processes simultaneously making data accesses on a set (upper left) where an underlying pairwise correlation exists between related data. Our method embeds the elements into a N-dimensional continuous space (lower) based on the dependency shown in the sequence.

4 EXPERIMENTS

To evaluate the utility of the correlation function learned by TAE, we compare it to other approaches on a simplified prefetching scenario. To do this, we simply retrieve the N most correlated elements for each location on a sequence, and then evaluate the frequency of their occurrence in the near future. The experiments are performed both on simulated and real computer storage access traces.

4.1 Synthetic Data

To generate synthetic data, we simulate a computer storage scenario. Initially, a dataset $\mathbb{S}$ of N elements is generated, in which each of the elements is assigned an index. A correlation function $cor(a, b), a, b \in \mathbb{S}$ is imposed by assigning a uniformly sampled value between a number of randomly selected pairs and among consecutively indexed elements. Then a list of processes independently sample $\mathbb{S}$ based on the correlation function to produce a mixed sequence $\mathcal{D}$. The procedure for generating synthetic data is shown in Algorithm 4.

For our experiments, we generated four sets of N=5K, 10K, 20K, 50K elements respectively, in each of which neighboring elements and $10N$ pairs are set correlated. A mixed sequence of length $100N$ is sampled from each set.

Algorithm 4 Mixed Sequence Generator

Given set $\mathbb{S}$, correlation function $cor(\cdot, \cdot)$, number of processes M, length of sequence T, process switch probability p_{switch}, process reinitialize probability p_{reinit}
$\mathcal{A} \leftarrow \text{Uniform}(M; \mathbb{S})$
$a \leftarrow \text{Uniform}(1; M)$
for $t \leftarrow 1, T$ **do**
  if $\text{Uniform}() \leq p_{\text{switch}}$ **then**
    $a \leftarrow \text{Uniform}(1; M)$
  end if
  if $\text{Uniform}() \leq p_{\text{reinit}}$ **then**
    $\mathcal{A}_a \leftarrow \text{Uniform}(1; \mathbb{S})$
  else
    $\mathcal{A}_a \leftarrow \text{Sample}(1; \mathbb{S}, cor(\mathcal{A}_a, \cdot))$
  end if
  $\mathcal{D}_t \leftarrow \mathcal{A}_a$
end for
return $\mathcal{D}$

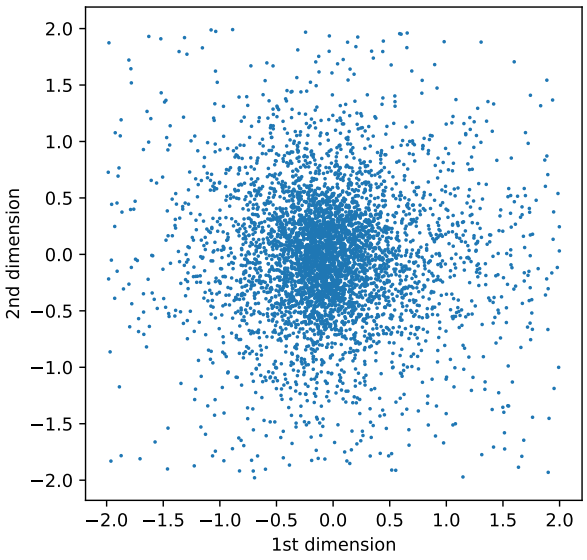

Figure 2: A 2D visualization of the embedding discovered by TAE on synthetic data, Attraction forces concentrate some of the data in the center, while the repulsion forces spread the rest of the data across the space.

4.2 Computer Storage Traces

We also apply TAE to computer storage I/O trace datasets, Financial1 and Financial2, from the UMass Trace Repository. These datasets were gathered from I/O traces of Online Transactional Processing systems running at two large financial institutions. The logical block address (LBA) information is used as the index, and the addressed storage unit is treated as the element. Because of the spatial locality and the distributed characteristics of computer storage, we assume that the correlation function on storage units is similar to that generated by our mixed sequence generator. Moreover, contemporary computers and servers are usually running a large number of threads simultaneously. Therefore, the real-world data access traces are both mixed and noisy, which is substantially different from many natural language processing sequence learning tasks.

Financial1 is a mixed sequence of length 5334987, which contains 710908 distinct units. Financial2 is a mixed sequence of length 3699195, which contains 296072 distinct units. As no higher-level information, e.g. file distribution or OS thread id, is provided with those datasets, we do not know the ground truth correlation function information for these traces.

4.3 Baseline Approaches

We compare TAE to the following techniques:

(1) **Informed Guess**. This heuristic simply makes random guesses informed by the prior probabilities of the elements, thus leveraging the degree of data concentration.

Kernel	linear
Window Size	64
Alpha	0.5
Beta	5e-7
Gamma	2
Number of Dimensions	8
Stopping Criterion	5e-3

Table 1: TAE Hyperparameters

(2) **Local Neighbors**. Prediction is made by by selecting a neighbor based on a Gaussian distribution, thus leveraging the degree of data locality.

(3) **Embedding LSTM** [4]: This approach learns the differences in computer storage addresses by assuming that the patterns are spatially invariant. It encodes the most frequent deltas (differences) into one-hot bins and then learns the sequence using a LSTM. They also propose an advanced approach combining clustering and LSTM. The advanced approach is designed to handle the long thread-switching interval of computer memory, which does not exist in computer storage. For this comparison, we use the same configuration as [4], which includes a 128×2 LSTM, 500k training steps, a sentence length of 64, embedding size of 128, and is restricted to the 50,000 most frequently appearing deltas.

(4) **Transition Matrix**. This technique predicts the most likely next element based on transition probabilities. Rather than using consecutive elements, we augment the transition probability of any two elements appearing within a short window. Conceptually, this approach is the most similar to TAE.

Table 1 shows the configuration of TAE used for our experiments. Based on our initial pilot studies, we concluded that its performance is not very sensitive to modifications in the hyperparameters.

4.4 Results

In each experiment, the first 70% of the sequence is used for training, and the last 30% is reserved for testing. The experimental results are reported both on the probability (Table 2) and the occurrence (Table 3) of the top-1 prediction in a window of the following 128 time steps. Because of the mixing, consecutive elements in a sequence are unlikely to be generated by the same process. Therefore, we count those as successful predictions provided they appear within a short future window.

Our results show that the mixed synthetic sequences are very challenging and defy the commonly used concentration and locality heuristics. Those heuristics (Informed Guess and Local Neighbors) perform better with the financial traces, which exhibit a higher skewness of the distribution and a stronger locality. As we expected, the embedding LSTM [4], which performs well in computer memory scenarios, is not suitable for mixed sequences that contain large numbers of elements. The LSTM does outperform the other methods on the Financial2 dataset. However the training of a LSTM network costs dozens of GPU hours, compared to a few minutes for TAE.

Surprisingly, our approach significantly outperforms transition matrix, which uses the same underlying idea of strengthening the

Approach\Dataset	Synthetic (5K)	Synthetic (10K)	Synthetic (20K)	Synthetic (50K)	Financial1	Financial2
Informed Guess	0.0215	0.0109	0.0053	0.0021	0.0479	0.0328
Local Neighbors	0.0441	0.0330	0.0287	0.0256	0.0612	0.0901
Embedding LSTM	0.0267	0.0219	0.0132	0.0103	0.1368	**0.2417**
Transition Matrix	0.2036	0.1979	0.1955	0.1961	0.2476	0.1801
TAE	**0.2708**	**0.2621**	**0.2586**	**0.2580**	**0.3239**	0.1947

Table 2: The probability that top-1 prediction appears in the following 128 time steps.

Approach\Dataset	Synthetic (5K)	Synthetic (10K)	Synthetic (20K)	Synthetic (50K)	Financial1	Financial2
Informed Guess	0.0264	0.0134	0.0065	0.0027	0.0980	0.0532
Local Neighbors	0.0526	0.0390	0.0335	0.0296	0.0916	0.1215
Embedding LSTM	0.0360	0.0300	0.0181	0.0150	0.2500	0.3469
Transition Matrix	0.2855	0.2800	0.2745	0.2764	0.5298	**0.4322**
TAE	**0.3827**	**0.3677**	**0.3610**	**0.3610**	**0.6711**	0.4249

Table 3: The average occurrences of top-1 prediction in the following 128 time steps.

correlation between elements that appear within a short time window. We believe that the transition matrix fails to model the indirect correlation between elements which is captured by our technique. Since TAE learns an embedding, it is easier to combine with other machine learning methods. It is also worth noting that our approach has a similar prediction accuracy across synthetic datasets of varying sizes, which demonstrates its scalability.

5 DISCUSSION

This section discusses possible extensions of Temporally Aware Embedding to handle unseen data elements along with a new correlation-based sampling technique.

5.1 Mutual Mapping

Only the embedding of $a \in \mathbb{S}'$ is meaningful with Temporally Aware Embedding. However, if there exists another meaningful encoding, e.g. a representation for a localization property in $\mathbb{S}$, the embedding can be extended to any $a \in \mathbb{S}$ through a mutual mapping between those two representations using continuous function approximators. Such an encoding exists for certain problems. For example, the addresses in computer systems are commonly encoded as tuples of bits, subsets of which may contain segment or page information.

Assume $u(a) \in \mathbb{U}$ is a proper encoding for any $a \in \mathbb{S}$. Two learning models, specifically deep neural networks, $\mathcal{F}(v_a) = u(a)$ and $\mathcal{G}(u(a)) = v_a$ can be trained respectively with the encoding and embedding pairs of $x \in \mathbb{S}'$. Based on continuity restriction, $\mathcal{F}$ is able to map every point v_a to $\mathbb{U}$ as a sample that is both temporally and locally correlated to nearby $b \in \mathbb{S}'$. Meanwhile, $\mathcal{G}$ allows us to discard the space-consuming embedding table. More importantly, it can embed unknown $a \in \mathbb{S}$ as long as $u(a)$ can be acquired.

5.2 Correlation-based Sampling

Since mutual mapping discards the table and extends the embedding, we propose a sampling method that can be extended to situations where $a \notin \mathbb{S}'$ or when some elements that are highly correlated to a do not appear in $\mathbb{S}'$.

Our sampling method starts by acquiring the embedding for a, which is $v_a = \mathcal{G}(u(a))$. Afterwards, a set of candidates $\{v_{b_1}, ..., v_{b_N}\}$ is sampled in $\mathbb{V}$ from a Gaussian distribution with $\mu = v_a$ and preselected small $\sigma \in \mathbb{R}$. We can obtain $u(b_i) = \mathcal{F}(v_{b_i})$ for $i = 1, ..., N$, and find the corresponding elements $b_1, ..., b_N$ with them if u is invertible. The procedure is as follows:

$$u(b_i) = \mathcal{F}(\mathcal{N}(\mathcal{G}(u(a)), \sigma^2)) \text{ for } i = 1, ..., N \tag{3}$$

According to mutual mapping, the sampled elements are neighbors to a both in $\mathbb{V}$ and $\mathbb{U}$. Thus they are highly correlated to a.

6 CONCLUSION AND FUTURE WORK

This paper introduces a technique, Temporally Aware Embedding (TAE), for learning correlation functions from mixed data sequences. The intuition behind TAE is that an embedding can be learned from element co-occurrences within a sliding time window. Our technique is simple and fast, which makes it highly suitable for computer architecture applications such as prefetching. It is robust to noisy and mixed data sequences, and also scales to datasets with large numbers of unique elements.

We evaluated TAE vs. a set of popular heuristics and deep learning methods. Even though the TAE learning rule is based largely on first order Markov statistics, it significantly outperforms a transition model created from a direct estimate of those statistics. We believe that the TAE embedding is more effective at capturing indirect relationships that are missed by the transition matrix. For instance, two elements that co-occur with another shared element, yet do not appear together, are not considered correlated by the transition matrix but are likely to appear in close proximity in our embedding. In future work, we plan to extend our embedding and sampling techniques to handle unseen data elements.

7 ACKNOWLEDGMENTS

This work was partially supported by Lockheed Martin Co.

REFERENCES

[1] [n. d.]. UMass Trace Repository. In *http://traces.cs.umass.edu/*.

[2] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. 2017. A survey of sequential pattern mining. *Data Science and Pattern Recognition* 1, 1 (2017), 54–77.

[3] John Gubner. 2006. *Probability and Random Processes for Electrical and Computer Engineers.* Cambridge University Press.

[4] Milad Hashemi, Kevin Swersky, Jamie A. Smith, Grant Ayers, Heiner Litz, Jichuan Chang, Christos Kozyrakis, and Parthasarathy Ranganathan. 2018. Learning Memory Access Patterns. In *arXiv:1803.02329*.

[5] John Hennessy and David Patterson. 2017. *Computer Architecture: A Quantitative Approach.* Elsevier.

[6] Zhenmin Li, Zhifeng Chen, Sudarshan M. Srinivasan, and Yuanyuan Zhou. 2004. C-Miner: Mining Block Correlations in Storage Systems. In *Proceedings of the USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA, 173–186. http://dl.acm.org/citation.cfm?id=1096673.1096695

[7] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. LSTM neural networks for language modeling. In *Annual Conference of the International Speech Communication Association.*

[8] I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems* (2014).