Extracting Stochastic Machines from Recurrent Neural Networks Trained on Complex Symbolic Sequences

Peter Tiňo
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
Email: petert@ai.univie.ac.at

Vladimir Vojtek
Department of Computer Science and Engineering
Slovak University of Technology
Ilkovicova 3, 812 19 Bratislava, Slovakia

Neural Network World, 8(5), pp. 517-530, 1998.

Abstract

We train recurrent neural network on a single, long, complex symbolic sequence with positive entropy. Training process is monitored through information theory based performance measures. We show that although the sequence is unpredictable, the network is able to code the sequence topological and statistical structure in recurrent neurons' activation scenarios. Such scenarios can be compactly represented through stochastic machines extracted from the trained network. Generative models, i.e. trained recurrent networks and extracted stochastic machines, are compared using entropy spectra of generated sequences. In addition, entropy spectra computed directly from the machines capture generalization abilities of extracted machines and are related to machines' long term behavior.

1 Introduction

Researchers have been attracted to the problem of finite state machine (FSM) inference with recurrent neural networks (RNNs) for a long time [2, 4, 5, 11, 13, 15]. RNNs were trained on relatively short sequences and then tested on much longer words to asses the quality of induced temporal structure within the trained networks. It was observed that recurrent neurons' activations tent to group in clusters reflecting attractive sets inside the network state space. Cluster detection techniques enabled representation of RNN behavior through FSM metaphor [10]. Direct comparison of the minimal forms of extracted machines and the machines used to generate training examples yielded the degree of training success.

The problem we study in this paper is of different nature. RNN is shown a long symbolic sequence S of positive entropy, i.e. difficult to predict. The network is trained to predict the next symbol given the current symbol and the history of previous symbols coded in recurrent neurons' activations. Since the training sequence S is unpredictable, the usual maximum and mean squared error based training criteria cannot be used. In other words, RNN will never learn to perform the prediction task on S perfectly. But the network still extracts a lot of useful information! During the training, we use information theory based measures to evaluate RNN as a model of S. After the training, we detect recurrent neurons' activation clusters with Kohonen self-organizing map (SOM) [8] and formulate RNN activity through stochastic machines (SMs).

To study and compare different generative models – trained RNNs and extracted SMs – we use statistical mechanical metaphor of entropy spectra. Entropy spectra of sequences unveil their statistical structure. Different formal temperatures accentuate different probability levels of subsequences. Entropy spectra computed directly from SMs capture the machines' long term behavior and are sequence and block length independent entities.

2 Preliminaries

We consider sequences $S = s_0 s_1 s_2 ...$ over a finite alphabet $\mathcal{A}$ with A elements generated by stationary information sources. For $i \leq j$, the string $s_i ... s_j$ is denoted by S_i^j, with $S_i^i = s_i$. Usually, the statistical structure of S is studied using a "sliding window" $w = w_1 ... w_n$ of length n. One determines the (empirical) probabilities $P_n(w)$ of finding a particular window w in S, if a block of n symbols (an n-block) is randomly chosen. A measure of n-block uncertainty is given by the block entropy

$$H_n = H(P_n) = - \sum_{w \in \mathcal{A}^n} P_n(w) \log P_n(w). \tag{1}$$

A measure of predictability of an added symbol (independent of block length) is then

$$h = \lim_{n \to \infty} \frac{H_n}{n} = \lim_{n \to \infty} h_n. \tag{2}$$

Entropy provides only a partial information concerning the sequence distribution P. A more fulfilling description is obtained through a spectrum of entropy measures. The spectrum is constructed using a formal parameter β that can be thought of as the inverse temperature in the statistical mechanics of spin systems [3]. The original n-block distribution $P_n(w)$ is transformed to the "twisted" distribution [14]

$$G_{\beta,n}(w) = \frac{P_n^\beta(w)}{\sum_{w \in \mathcal{A}^n} P_n^\beta(w)}. \tag{3}$$

The most probable and the least probable n-blocks of the original distribution $P_n(w)$ become dominant in the positive zero and the negative zero temperature regimes, $G_{\infty,n}(w)$ and $G_{-\infty,n}(w)$ respectively. Varying β from 0 to ∞ amounts to a shift from all allowed n-blocks to the most probable ones by accentuating still more and more probable subsequences. Varying β from 0 to $-\infty$ accentuates less and less probable n-blocks with the extreme of the least probable ones.

Thermodynamic entropy density is approximated by

$$h_{\beta,n} = \frac{- \sum_{w \in \mathcal{A}^n} G_{\beta,n}(w) \log G_{\beta,n}(w)}{n} \tag{4}$$

and is given asymptotically by

$$h_\beta = \lim_{n \to \infty} h_{\beta,n}. \tag{5}$$

SMs are much like non-deterministic finite state machines except that the state transitions take place with probabilities prescribed by a distribution T. To start the process, the machine M chooses the initial state according to the "initial" distribution π, and then, at any given time step after that, the machine is in some state $i \in Q$, and at the next time step moves to another state $j \in Q$ outputting some symbol $s \in \mathcal{A}$, with the transition probability $T_{i,j,s}$.

Denote the transition matrix associated with a symbol s and the stochastic state transition matrix by $T(s)$ and $\mathcal{T}$ respectively

$$T(s)_{i,j} = T_{i,j,s} \quad \forall i,j \in Q \text{ and } \forall s \in \mathcal{A}, \tag{6}$$

$$\mathcal{T} = \sum_{s \in \mathcal{A}} T(s). \tag{7}$$

Ignoring the state transition labels, $\mathcal{T}$ describes a Markov chain over the machine states Q.

As in the previous section, we introduce parameterized transition probabilities $T^{\beta}_{i,j,s}$ and think of each setting of the formal parameter β as emphasizing a different set of sequences generated by M. The state transition matrix becomes ($\mathcal{T}_\beta$ is no longer a stochastic matrix)

$$\mathcal{T}_\beta = \sum_{s \in \mathcal{A}} (T(s))^{\beta}. \tag{8}$$

Denote the left and right eigenvectors of $\mathcal{T}_\beta$ associated with the maximum eigenvalue λ_β by v^L_β and v^R_β respectively. v^L_β and v^R_β are chosen so that their dot product is unity. The equivalent stochastic process with transition probabilities weighted according to $\mathcal{T}_\beta$ is given by the stochasticized version [14]

$$(\mathcal{R}_\beta)_{ij} = \frac{(\mathcal{T}_\beta)_{ij} \left(v^R_\beta\right)_j}{\lambda_\beta \left(v^R_\beta\right)_i}. \tag{9}$$

The metric entropy of the parameterized stochastic machine M_β,

$$h_\mu(\beta) = -\sum_{i,j \in Q} p_{\beta,i} (\mathcal{R}_\beta)_{ij} \log (\mathcal{R}_\beta)_{ij} \tag{10}$$

is an average of transition uncertainty over all machine states. In the deterministic case, when for every state, each symbol s uniquely determines the next state, $h_\mu(\beta)$ is also the thermodynamic entropy density h_β (eq. (5)) of sequences generated by M [14].

There is an ongoing discussion concerning complexity measures of time-varying patterns [6]. In this study a sequence S is considered "complex" if it appears to be random, i.e. the entropy rate h (eq. (2)) is positive, and can be faithfully modeled only with nontrivial stochastic machines. "Nontriviality" of a machine M is measured by its topological complexity $C_0 = \log |Q|$ [3].

3 Neural models

The RNN presented in figure 1 was shown to be able to learn mappings that can be described by finite state machines [11].

A one-of-A encoding of symbols from the alphabet $\mathcal{A}$ is used with one input and one output neuron, $I^{(t)}_i$ and $O^{(t)}_i$ respectively, devoted to each symbol. There are two types of second order hidden neurons in the network. K hidden non-recurrent neurons $H^{(t)}_k$ and L hidden recurrent neurons $R^{(t)}_l$ called the state neurons.

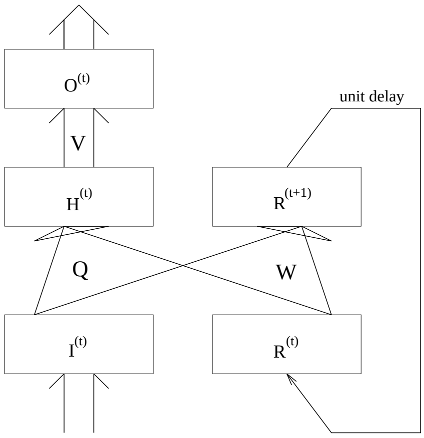

Figure 1: RNN architecture

W_{iln}, Q_{jln} and V_{mk} are real-valued weights and g is a sigmoid function $g(x) = 1/(1+e^{-x})$. The activations of hidden non-recurrent neurons are determined by

$$H_j^{(t)} = g\left(\sum_{l,n} Q_{jln} R_l^{(t)} I_n^{(t)}\right).$$

The activations of state neurons at the next time step $(t+1)$ are computed as follows:

$$R_i^{(t+1)} = g\left(\sum_{l,n} W_{iln} R_l^{(t)} I_n^{(t)}\right).$$

The network output is determined by

$$O_m^{(t)} = g\left(\sum_{k} V_{mk} H_k^{(t)}\right).$$

The *state* and *association networks* consist of layers $I^{(t)}$, $R^{(t)}$, $R^{(t+1)}$ and $I^{(t)}$, $R^{(t)}$, $H^{(t)}$, $O^{(t)}$ respectively. The architecture strictly separates the input history coding part (state network) from the part responsible for the association of the so far presented inputs[1] with the network output (association network).

The network is trained with RTRL [7] on a single, long symbolic sequence $S = s_0 s_1 s_2 ...$, to predict, at each point in time, the next symbol.

To start the training, the initial network state $R^{(0)}$ (activations of recurrent neurons at time 0) is randomly generated. The network is reset with $R^{(0)}$ at the beginning of each training epoch.

After the training, the network is seeded with the initial state $R^{(0)}$ and the first training symbol s_0. For the next T_1 "pre-test" steps, based on the current input and state, the next network state is computed, and it comes into play, together with the next symbol from S, at the next time step. This way, the network is given a right "momentum" in the state path starting in the initial "reset" state $R^{(0)}$.

After T_1 pre-test steps, the network generates a symbol sequence by itself. In each of T_2 test steps, the network output is interpreted as a new symbol that will appear at the net input at the next time step. The network state sequence is generated as before. The output activations

[1] the last input $I^{(t)}$, together with a code $R^{(t)}$ of the recent history of past inputs

$O^{(t)} \in (0,1)$ are transformed into "probabilities" $P_i^{(t)}$,

$$P_i^{(t)} = \frac{O_i^{(t)}}{\sum_{j=1}^{A} O_j^{(t)}}, \quad i = 1, 2, ..., A, \tag{11}$$

and the new symbol $\hat{s}^{(t)} \in \mathcal{A}$ is generated with respect to the distribution $P_i^{(t)}$ (the number of output neurons is equal to the number of symbols in $\mathcal{A}$).

We assume that the reader is familiar with the standard unsupervised training procedure for SOM [8]. SOM places a fixed number of codevectors w_i (codevectors) into the map input space Y, subject to a minimum distortion constraint. w_i represents a part of the input space,

$$V(i) = \left\{ y \in Y |\ d(y, w_i) = \min_j \{d(y, w_j)\} \right\},$$

where d is the Euclidean distance. $V(i)$ is referred to as the Voronoi compartment of the codevector i.

4 Extracting stochastic machiness from trained RNNs

The stochastic machine $M_{RNN} = (Q, \mathcal{A}, T, \pi)$ is extracted from the RNN trained on a sequence $S = s_0 s_1 s_2 ...$ using the following algorithm:

1. Quantize the RNN state space by running a Kohonen SOM on RNN states recorded during the RNN testing.
2. The unique initial state is a pair (s_0, i_0), where i_0 is the index of the Kohonen unit defining the Voronoi compartment $V(i_0)$ containing the network "reset" state $R^{(0)}$, i.e. $R^{(0)} \in V(i_0)$. Set $Q = \{(s_0, i_0)\}$.
3. For T_1 pre-test steps $1 \leq t \leq T_1$
   - $Q := Q \cup (s_t, i_t)$, where $R^{(t)} \in V(i_t)$
   - add the edge[2] $(s_{t-1}, i_{t-1}) \rightarrow^{s_t} (s_t, i_t)$ to the topological skeletal state-transition structure of M_{RNN}.
4. For T_2 test steps $T_1 < t \leq T_1 + T_2$
   - $Q := Q \cup (\hat{s}_t, i_t)$, where $R^{(t)} \in V(i_t)$ and $\hat{s}_t$ is the symbol generated at the RNN output.
   - add the edge[3] $(\hat{s}_{t-1}, i_{t-1}) \rightarrow^{\hat{s}_t} (\hat{s}_t, i_t)$ to the set of allowed state-transitions in M_{RNN}.

The probabilistic structure is added to the topological structure of M_{RNN} by counting, for all state pairs $(p, q) \in Q^2$ and each symbol $s \in \mathcal{A}$, the number $N(p, q, s)$ of times the edge $p \rightarrow^s q$ was invoked while performing steps 3 and 4. The state-transition probabilities are then computed as

$$T_{p,q,s} = \frac{N(p, q, s)}{\sum_{r \in Q, a \in \mathcal{A}} N(p, r, a)}. \tag{12}$$

The philosophy of the extraction procedure is to let the RNN act as in the testing mode, and interpret the activity of RNN, whose states have been factorized into a finite set of clusters, as a stochastic machine M_{RNN}.

When T_1 is equal to the length of the training set minus 1, and $T_2 = 0$, i.e. when RNN is driven with the training sequence, we denote the extracted machine by $M_{RNN(S)}$. Otherwise, the construction is stochastic and one gets a little different machine M_{RNN} each time the procedure is run.

[2] from the state (s_{t-1}, i_{t-1}) to the state (s_t, i_t), labeled with s_t

[3] $\hat{s}_{T_1} = s_{T_1}$

5 Experiments

For a stationary ergodic process that has generated a sequence S of length N, the Lempel-Ziv codeword length for S, divided by N, is computationally efficient and reliable estimate $h_{LZ}(S)$ of the source entropy [16]. In particular, let $c(S)$ denote the number of phrases in S resulting from the incremental parsing of S, i.e. sequential parsing of S into distinct phrases such that each phrase is the shortest string which is not a previously parsed phrase. The Lempel-Ziv codeword length for S is approximated with $c(S)\log c(S)$ [16]. Lempel-Ziv approximation of the source entropy is then

$$h_{LZ}(S) = \frac{c(S)\log c(S)}{N}. \tag{13}$$

The notion of "distance" between distributions used in this paper is a well-known measure in information theory, called Kullback-Leibler divergence. It is also known as the relative, or cross entropy. Let P and Q be two Markov probability measures, each of some (unknown) finite order. The divergence between P and Q is defined by

$$d^{KL}(Q|P) = \limsup_{n\to\infty} \frac{1}{n} \sum_{w\in\mathcal{A}^n} Q_n(w) \log \frac{Q_n(w)}{P_n(w)}.$$

d^{KL} measures the expected additional code length required when using the ideal code for P instead of the ideal code for the "right" distribution Q.

Suppose we have only length-N realizations S_P and S_Q of P and Q respectively. Analogically to Lempel-Ziv entropy estimation, there is a procedure for determining $d^{KL}(Q|P)$ from S_P and S_Q [16]. The procedure is based on Lempel-Ziv sequential parsing of S_Q with respect to S_P: First, find the longest prefix of S_Q that appears in S_P, i.e. the largest integer m such that the m-blocks $(S_Q)_0^{m-1}$ and $(S_P)_i^{i+m-1}$ are equal, for some i. $(S_Q)_0^{m-1}$ is the first phrase of S_Q with respect to S_P. Next, start from the mth position in S_Q and find, in a similar manner, the longest prefix $(S_Q)_m^k$ that appears in S_P, and so on. The procedure terminates when S_Q is completely parsed with respect to S_P. Let $c(S_Q|S_P)$ denote the number of phrases in S_Q with respect to S_P. Then, the Lempel-Ziv estimate of $d^{KL}(Q|P)$ is computed as [16]

$$d^{KL}_{LZ}(S_Q|S_P) = \frac{c(S_Q|S_P)\log N}{N} - h_{LZ}(S_Q).$$

In the experiment, we used Santa Fe competition data recorded from a laser in a chaotic state available on the Internet (`http:// www.cs.colorado.edu/ ~andreas/ Time-Series/ SantaFe.html`).

Time series of approximately 10,000 points was transformed into a symbolic sequence over $\{a, b, c, d\}$ by partitioning the signal range into 4 regions $[0, 50)$, $[50, 200)$, $[-64, 0)$ and $[-200, 64)$. The regions were determined by close inspection of the data and correspond to clusters of low and high positive/negative laser activity changes.

We trained two RNNs with 2 and 5 recurrent (state) neurons. The two RNNs are referred to as RNN_2 and RNN_5 respectively. The training process consisted of 10 runs with 100 passes through S in each run.

During the training, we used let the RNNs to generate sequences $S(RNN)$ of length equal to the length of the training sequence S and computed the corresponding entropy and cross entropy estimates $h_{LZ}(S(RNN))$ and $h^{KL}_{LZ}(S|S(RNN))$, respectively. Figures 2 and 3 show the summary of entropy measures for the training process.

On average, the 5 state neuron network RNN_5, did better than its 2 state counterpart RNN_2, because RNN_5 developed more sophisticated dynamical representations of the temporal structure in S, than RNN_2. A consequence of a more powerful potential for developing dynamical scenarios in RNN_5 is a greater liability to overfitting, as seen in figure 3. Relatively simple dynamical

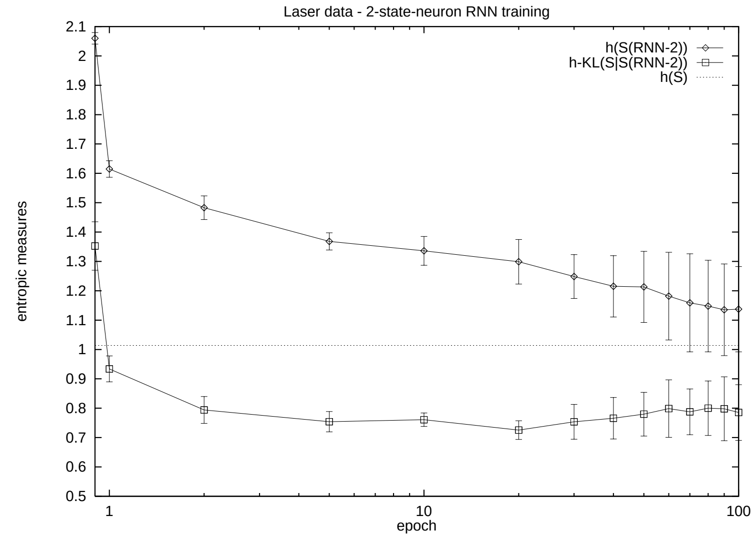

Figure 2: Training of 2-state-neuron RNN on laser data S. Shown are the mean values and standard deviations of the L-Z entropy estimates $h(\cdot)$ and numbers of cross-phrases $c(S|\cdot)$ (scaled by 10^{-3}) across 10 sequence generation realizations. The horizontal Dotted line corresponds to the L-Z estimate of the training sequence entropy. We use the number of phrases $c(S|\cdot)$ in the training sequence S with respect to model generated sequence instead of cross entropy estimate $h^{KL}_{LZ}(S|\cdot)$. $h^{KL}_{LZ}(S|\cdot)$ is completely determined by $c(S|\cdot)$ and $h_{LZ}(S)$. Due to the finite sequence length effects, the cross entropy estimates can be negative. The higher is the number of phrases $c(S|S(\mathcal{M}))$, the bigger is the estimated statistical distance between sequences S and $S(\mathcal{M})$.

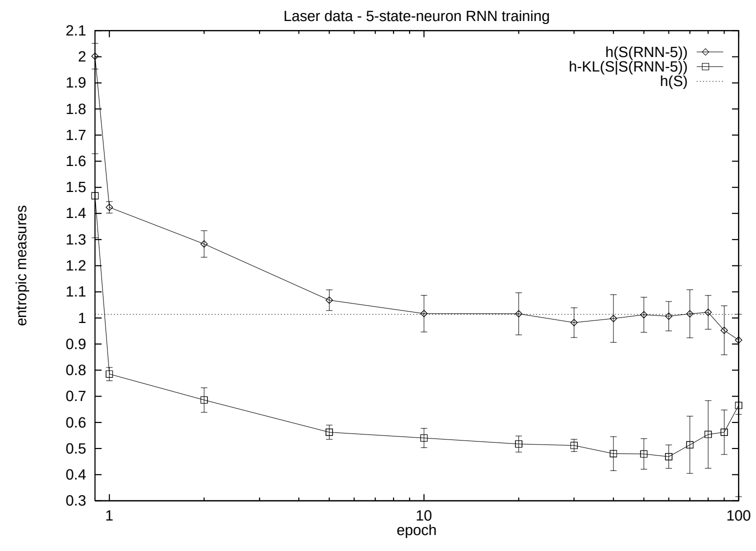

Figure 3: Training of 5-state-neuron RNNs on laser data S. The experimental setting is explained in the caption to the previous figure.

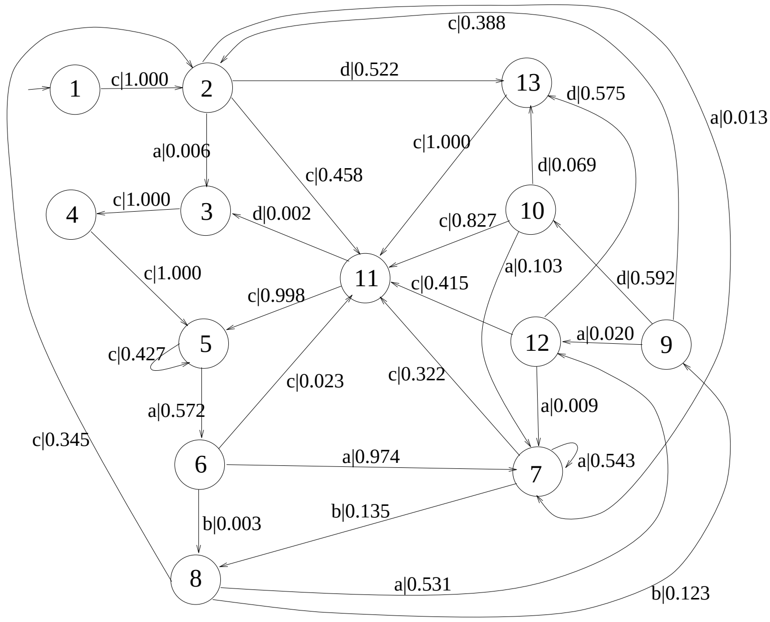

Figure 4: The machine M_{RNN_2} extracted from a 2-state-neuron RNN trained on laser data.

regimes in RNN_2 - for each input symbol, attractive fixed points, or period-two orbits - resulted in oversimplifying dynamical patterns of allowed n-blocks, and RNN_2 systematically overestimated the entropy of S.

After training, we picked up representative 2 and 5 state neuron solutions based on their entropy measures performance. In the test mode ($T_1 = 10$ and T_2 is equal to the training sequence length), there were respectively 4 and 9 dominant clusters in the state space of the two RNN representatives RNN_2 and RNN_5. The clusters were detected using Kohonen SOM. The machines M_{RNN_2}, $M_{RNN_2(S)}$, and M_{RNN_5},$M_{RNN_5(S)}$ had 13 and 17 states respectively with deterministic state transition structure. As an example, the machine M_{RNN_2} can be seen in figure 4. The initial state is marked with an incoming arrow. The transitions are labeled with probabilities and associated symbols.

The generative models RNN_2, $M_{RNN_2(S)}$, M_{RNN_2}, RNN_5, $M_{RNN_5(S)}$ and M_{RNN_5} were compared by first letting each model $\mathcal{M}$ generate a sequence $S(\mathcal{M})$ equal in length to the training sequence S. Then, the entropy spectra h_β (eq. (5)) of $S(\mathcal{M})$ were estimated from 6-block distributions[4] through $h_{\beta,6}$ (eq. (4)). The spectra are shown in figure 5.

RNN_5 and the corresponding extracted machines $M_{RNN_5(S)}$, M_{RNN_5} produce the least number of allowed 6-blocks (compare their topological entropies $h_{0,6}(S(\cdot))$ with those of other models - see table 1). High probability 6-blocks generated by them do not contain much probabilistic structure. This is also true for high probability 6-blocks generated by $M_{RNN_2(S)}$. Hence, as positive temperature is lowered, the spectra decrease only gradually.

A stronger probabilistic structure in high probability 6-blocks then that observed in S is present in sequences generated by RNN_2 and its associated machine M_{RNN_2}.

Due to deterministic state transition topology in the extracted machines, one can compute thermodynamic entropy spectra h_β directly from their markovian state transition structure (eq. (10)). In figure 6 we see how far from the true spectra h_β are their 6-block approximations $h_{\beta,6}$. For high probability sequences, M_{RNN_2} and M_{RNN_5} define similarly probabilistically structured sequence

[4] There are 4^6 possible 6-blocks. Since the length of training sequence is approximately 10,000 and its topological entropy is close to 1, we found 6 a reasonable choice for the block length.

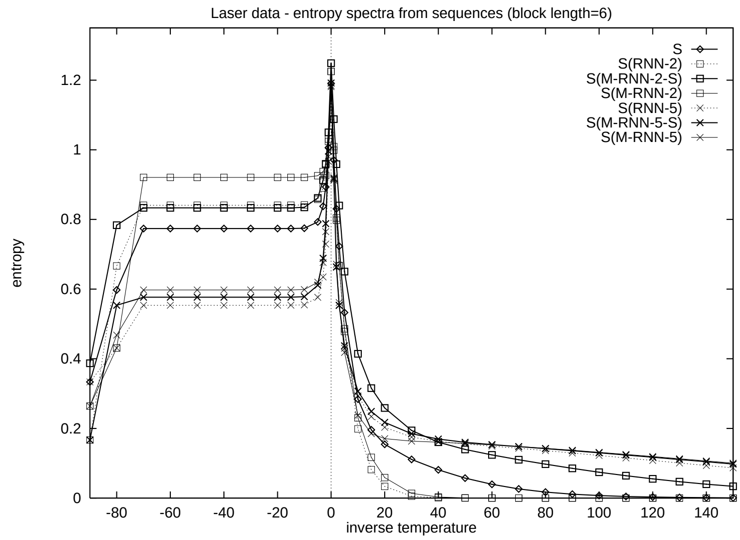

Figure 5: Estimation of entropy spectra (based on 6-block statistics) of sequences generated by the source models RNN_2, $M_{RNN_2(S)}$, M_{RNN_2}, RNN_5, $M_{RNN_5(S)}$ and M_{RNN_5}. The labels for the models are `RNN-2, M-RNN-2-S, M-RNN-2, RNN-5, M-RNN-5-S` and `M-RNN-5` respectively. `S` stands for the training sequence.

distributions with entropy profiles close to 2-recurrent neuron 6-block profiles $h_{0,6}(S(RNN_2))$ and $h_{0,6}(S(M_{RNN_2}))$. A more loose probabilistic structure of high probability 6-blocks is visible for the machines $M_{RNN_2(S)}$ and $M_{RNN_5(S)}$. It is surprising, how strong a probabilistic structure is produced by M_{RNN_2}, $M_{RNN_2(S)}$ and $M_{RNN_5(S)}$ for low probability sequences. They generate only a few of them. To unveil this structure with block-based statistics, we would need much longer sequence and block lengths. There is a set of very low probability sequences produced by M_{RNN_5} with a very little probabilistic structure, causing the machine spectrum of M_{RNN_5} to decrease only slowly with increasing temperature in the negative range.

Table 1 compares generative models with respect to machine entropy estimates, and entropy measures on produced sequences. One sees that several measures are needed to evaluate the performance of training sequence candidate models. $h_{1,6}$, $h_\mu(1)$ and h_{LZ} measure the average uncertainty per symbol, while $h_{0,6}$ and $h_\mu(0)$ measure the diversity of allowed subsequences.

In the experiment, because of the finite sequence and short block lengths, the 6-block topological and metric entropy estimates $h_{0,6}$ and $h_{1,6}$ were higher than their machine counterparts $h_\mu(0)$ and $h_\mu(1)$ [14]. Finite sequence length is also responsible for the systematic overestimation of $h_{LZ}(\cdot)$.

6 Discussion and conclusion

We studied the problem of modeling long, complex symbolic sequences S with recurrent neural networks and stochastic machines. To this end, we used the recurrent network architecture with both first- and second-order neural units that was introduced in [11]. Since the traditional output-error-based performance criteria are useless in monitoring the training process on positive entropy sequences, we use information-theoretic measures instead. Majority of detectable information is gained during the first few training epochs.

After the training, Kohonen SOM quantizes RNN state space. The two neural networks then cooperate in creating a stochastic machine model M_{RNN} that mimics the RNN activity. In the process of building M_{RNN}, the RNN plays an active role by, based on what it has learned during

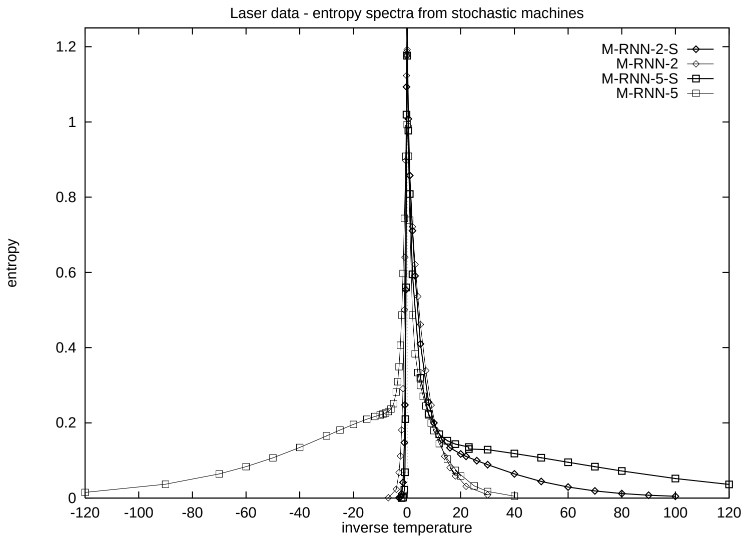

Figure 6: Machine entropy spectra of M_{RNN_2}, $M_{RNN_2(S)}$, M_{RNN_5} and $M_{RNN_5(S)}$.

the training, producing a symbolic sequence as in the test mode. The SOM "sits" inside the RNN and interprets the dynamical state representations of the RNN.

In order to interpret the generative potential of constructed models, it is useful to interpret symbolic sequences as lattice configurations of "spins" and study their statistical mechanics [9] [14]. Entropy spectra accentuate subsequences at different probability levels and can be directly computed from machines' deterministic state transition structure. Such spectra are independent of sequence and block lengths, and show what the machines represent in terms of long term behavior. In this manner, one can interpret the way in which a model generalized the training sequence. From the modeling point of view, the most important model characteristics are metric and topological information theoretic measures. Measures at other high positive/negative inverse temperatures reflect details in machine construction procedures and can be of importance only when generating much longer sequences than the training one. Finite length sequences do not contain sufficient information for determining extreme inverse temperature statistics.

Our experience is that the sequences $S(M_{RNN})$ generated by the extracted machines M_{RNN} not only share the same topological and metric entropies with their recurrent network counterparts $S(RNN)$, but also, their entropy spectra remain close to each other across different block lengths. This is particularly useful, since contrary to RNNs, the knowledge encoded in stochastic machines is amenable to formal analysis. One can employ automata theory to study topological structure of allowed subsequences, or statistical mechanics to concentrate on entropy spectra of subsequences at different probability levels. However, a note of caution is needed here: the measures at high positive/negative inverse temperatures reflect details in machine construction procedures and can be of importance only when generating sequences that are much longer than the training one. Finite length sequences do not contain sufficient information for determining extreme inverse temperature statistics.

In general, compared to RNNs and their finite state emulators M_{RNN}, the machines $M_{RNN(S)}$ constructed with the training sequence driven extraction have equal or better modeling performance. While the machines M_{RNN} are finite state representations of RNNs working autonomously in the test regime, the machines $M_{RNN(S)}$ are finite state 'snapshots" of the dynamics inside RNNs while processing the training sequence S. Long term behavior of a RNN working as a source is governed by attractive sets inside the network state space [1, 10]. Dominant attractive sets are

	$h_{0,6}$	$h_\mu(0)$	$h_{1,6}$	$h_\mu(1)$	$h_{LZ}(\cdot)$	$c(S\|\cdot)$
S	1.189	–	0.970	–	1.013	–
$S(RNN_2)$	1.224	–	1.009	–	1.211	725
$S(M_{RNN_2(S)})$	1.248	1.260	1.088	0.857	1.220	651
$S(M_{RNN_2})$	1.224	1.192	0.998	0.856	1.198	730
$S(RNN_5)$	1.181	–	0.920	–	1.012	479
$S(M_{RNN_5(S)})$	1.191	1.176	0.916	0.808	1.094	485
$S(M_{RNN_5})$	1.184	0.993	0.912	0.738	1.097	490

Table 1: Laser data. Comparison of generative models with respect to machine entropy estimates and entropy measures on produced sequences.

captured by the extraction procedure as frequently visited states and state transitions of the machine M_{RNN}. The same applies to the extraction of the machines $M_{RNN(S)}$, but this tendency is hampered by the training set driven external input. Probably, this makes the behavior of the machines $M_{RNN(S)}$ more distant from the modeling behavior of RNNs and more close to statistics on the training sequence.

Table 1 summarizes performance of the generative models with respect to machine entropy estimates and entropy measures on produced sequences.

In our experiments with other training sequences (e.g. binary sequences corresponding to logistic map generated time series quantized with respect to the critical point), we observed that the complexity of computational structure in training sequences was reflected also by dynamical state representations of trained RNNs, and hence by the complexity of extracted machines M_{RNN} [12].

Acknowledgments This work was supported by the VEGA grant 95/5195/605.

References

[1] M.P. Casey. The dynamics of discrete-time computation, with application to recurrent neural networks and finite state machine extraction. *Neural Computation*, 8(6):1135–1178, 1996.

[2] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1(3):372–381, 1989.

[3] J.P. Crutchfield and K. Young. Computation at the onset of chaos. In W.H. Zurek, editor, *Complexity, Entropy, and the physics of Information, SFI Studies in the Sciences of Complexity, vol 8*, pages 223–269. Addison-Wesley, 1990.

[4] S. Das and M.C. Mozer. A unified gradient–descent/clustering architecture for finite state machine induction. In J.D. Cowen, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 19–26. Morgan Kaufmann, 1994.

[5] C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun, and Y.C. Lee. Learning and extracting finite state automata with second–order recurrent neural networks. *Neural Computation*, 4(3):393–405, 1992.

[6] P. Grassberger. Information and complexity measures in dynamical systems. In H. Atmanspacher and H. Scheingraber, editors, *Information Dynamics*, pages 15–33. Plenum Press, New York, 1991.

[7] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation.* Addison–Wesley, Redwood City, CA, 1991.

[8] T. Kohonen. The self–organizing map. *Proceedings of the IEEE*, 78(9):1464–1479, 1990.

[9] J.L. McCauley. *Chaos, Dynamics and Fractals: an algorithmic approach to deterministic chaos.* Cambridge University Press, 1994.

[10] P. Tiňo, B.G. Horne, C.L. Giles, and P.C. Collingwood. Finite state machines and recurrent neural networks – automata and dynamical systems approaches. In J.E. Dayhoff and O. Omidvar, editors, *Neural Networks and Pattern Recognition*, pages 171–220. Academic Press, 1998.

[11] P. Tiňo and J. Sajda. Learning and extracting initial mealy machines with a modular neural network model. *Neural Computation*, 7(4):822–844, 1995.

[12] P. Tiňo and V. Vojtek. Modeling complex sequences with recurrent neural networks. In G.D. Smith, N.C. Steele, and R.F. Albrecht, editors, *Artificial Neural Networks and Genetic Algorithms*, pages – to appear. Springer Verlag Wien New York, 1998.

[13] R.L. Watrous and G.M. Kuhn. Induction of finite–state languages using second–order recurrent networks. *Neural Computation*, 4(3):406–414, 1992.

[14] K. Young and J.P. Crutchfield. Fluctuation spectroscopy. In W. Ebeling, editor, *Chaos, Solitons, and Fractals, special issue on Complexity*, 1993.

[15] Z. Zeng, R.M. Goodman, and P. Smyth. Learning finite state machines with self–clustering recurrent networks. *Neural Computation*, 5(6):976–990, 1993.

[16] J. Ziv and N. Merhav. A measure of relative entropy between individual sequences with application to universal classification. *IEEE Transactions on Information Theory*, 39(4):1270–1279, 1993.