Self-Organizing Map of Symbol Strings with Smooth Symbol Averaging

Panu Somervuo
Neural Networks Research Centre
Helsinki University of Technology
P.O.Box 5400, FIN-02015 HUT, Finland
tel. +358 9 451 3284, fax +358 9 451 3277
email: panu.somervuo@hut.fi

Keywords: dynamic time warping, self-organizing map, symbol string, string average

***Abstract*—** This work presents a method for smooth interpolation between symbol strings. The method is applied to the online training of the Self-Organizing Map with symbol strings as data.

1 Introduction

The Self-Organizing Map (SOM) [4, 6] is an unsupervised method for forming the representation of the data. It consists of local data models located at the nodes of the low-dimensional map grid. The models are adapted via competitive learning process.

There are two approaches for constructing the SOM for a new data set. One is to convert the data into feature vectors so that the comparison and averaging can be done using familiar Euclidean distance and arithmetic averaging. Another possibility is to modify the SOM algorithm itself. This is the approach taken in the current study, where the SOM is constructed for symbol string data.

Symbol strings are encountered in many application fields like speech recognition and bioinformatics. This work describes a method which allows the construction of the SOM for symbol strings with smooth symbol averaging. In earlier work with the SOM of symbol strings [7, 8, 9, 10, 16]. the data and the models in the SOM have been considered as discrete items. Quantized data and models may result that there are duplicate models on the SOM [10]. The proposed method for computing string average has no such problems. In this work, the models in the SOM are able to interpolate smoothly between discrete data clusters. Only when the soft symbol representations are converted back to discrete symbols after the SOM training, the duplicates may occur.

In this work the models in the SOM are vector sequences and the data strings are also represented in the similar form. Each symbol element of the data string is converted into a vector whose dimension corresponds to the size of the alphabet. The length of the vector sequence equals to the length of the original symbol sequence. Vector representation allows arithmetic averaging between the symbols and guarantees smooth behavior of the models in the SOM. Sequences with varying lengths are compared using dynamic time warping (DTW) [13, 14] and the average of vector sequences with unequal lengths is also well-defined [12, 14].

The construction of the SOM for variable-length feature vector sequences has been demonstrated earlier in [15]. The novelty in this work is the apply the methodology to the symbol strings. In addition, in this work all required formulae are expressed in detail.

The paper is organized as follows: Sec. 2 explains the background and related work to the SOM of symbol strings, Sec. 3 illustrates different string averages, Sec. 4 describes the SOM training with smooth string averages, and Sec. 5 illustrates the method with examples.

2 Previous work related to the SOM of symbol strings

In [7] it was for the first time shown that the SOM can be constructed for any data set for which a similarity or dissimilarity measure between its elements is defined. This was based on the use of the batch-SOM training and the definition of the generalized median as the model associated with each SOM node.

The first application of this method was the construction of the SOM for symbol strings [7, 8, 9]. Two types of medians were used: the set median, which is an existing element of the data set, and the median, which does not have to be an exact replica of any element in the data set. The set median is applicable to dense data sets, while the latter kind of median may interpolate sparse data better.

If set medians are used, the models are always copies of some data items. The benefit of this is that one algorithm can be applied without any modifications to the new data sets. But if the type of the data is restricted, e.g. the SOM is to be constructed for symbol strings as in the current work, then more data-type oriented interpolation methods can be used.

The SOMs can be trained using the online or batch algorithm. Batch algorithm for symbol string SOM has been used in [7, 8, 9, 10, 16] and online algorithms have been experimented in [1, 3]. In this work a new method is proposed for computing the string average. This method fits naturally to the online training algorithm.

3 Examples of string averages

Two approaches for computing the string average has been presented in [5]. One of them, the set median, is defined to be the string which has the smallest sum of distances to other strings in the set. The fine-tuned median string can then be obtained by systematically varying the symbols in the set median string. These methods have been utilized in the construction of the SOM of symbol strings in [7, 8, 9]. Further modifications for computing the string average has been proposed in [3].

If the data item can be converted into feature vector, the interpolation and averaging can be computed by arithmetic means. In [2] the symbol string data was clustered using the SOM. The data strings which represented protein sequences were first aligned and then converted into fixed-dimensional feature vectors. String alignment is a process where the strings are shifted to the left or right in respect of other strings so that the maximum mutual coherence between the strings is attained. The coherence is measured by the position-wise symbol similarity. In [2] sequences consisted of amino-acid symbols. Because of the alignment, strings could then be converted into fixed-dimensional feature vector sequences. Each symbol was converted into 20-dimensional vector (there were 20 symbols in the amino-acid alphabet) and the length of the vector sequence was the length of the global alignment of the entire string set. Since the length of the vector sequence was the same for all strings, Euclidean distance and arithmetic averaging could be used in the SOM training.

In [12], two clustering methods of DTW-templates were considered. The methods were applied to feature vector sequences which represented isolated word utterances. The representative sequence of the cluster, the minimax center, was defined to be the sequence which had the smallest maximum distances to other sequences in the cluster. That approach was called clustering without averaging (UWA). Another method was also presented, clustering with full averaging (UFA), where the averaged sequence was obtained averaging feature vectors of two sequences along their warping path.

The current work combines the two abovementioned approaches by utilizing the vector coding of the symbols [2] and dynamic time warping when computing the distances and averages between vector sequences [12].

Different string averages are illustrated in Fig. 1. Set median and median in Fig. 1 (a) and (b) are computed according to [5]. Vector-representation based average is shown in Fig. 1(c). Since there are seven symbols in the alphabet: A, B, C, E, L, S, and T, each symbol in the strings of Fig. 1(a) is converted into seven-dimensional vector. The details of the computation of the average sequence are described in Sec. 4.3. In Fig. 1(d), vector-sequence average of Fig. 1(c) is converted into quantized symbol string by selecting the symbol in each sequence position which corresponds to the largest vector element in that position. In this example the result is equal to the median string of Fig. 1(b).

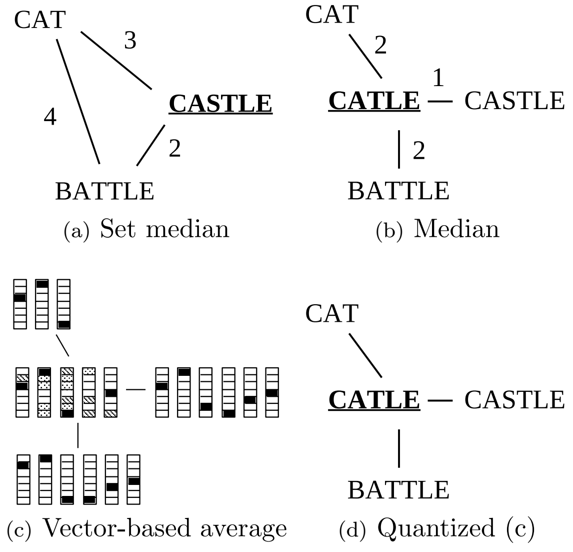


Figure 1: Examples of string averages for the data set containing three strings: “CAT”, “CASTLE”, and “BATTLE”. The numbers in (a) and (b) are Levenshtein distances between the strings. String average is shown underlined in each subfigure.

4 Online SOM algorithm for symbol strings

The SOM training consists of two steps which are iterated [4, 6]: 1) finding the best-matching unit (BMU) for the input and 2) adaptation of the models.

The index of the BMU to the given input X_t is defined as:

$$c(X_t) = \arg\min_k D(X_t, M_k), \tag{1}$$

where M_k denotes the model of the kth map node and $D(X_t, M_k)$ is the distance measure. Distance measure for variable-length vector sequences is defined in Sec. 4.2.

In accordance with the original SOM algorithm for vectors, the adaptation of the model M_k, which is now a *sequence* of vectors, has the following form in the online training:

$$M_k(t+1) = M_k(t) + h(c,k,t)[X_t - M_k(t)], \quad (2)$$

where $h(c,k,t)$ denotes the neighborhood function which determines the learning rates of the models, t denotes the training cycle. Symmetric Gaussian neighborhood function can be used whose center is located at the BMU, $c = c(X_t)$. The details of the computation of the weighted average between two vector sequences are given in Sec. 4.3.

4.1 Symbol string encoding

Symbol strings are encoded into vector sequences in the following way. Let $S_t = [s_{t1}s_{t2}\dots s_{tL}]$ denote the tth symbol string with length L. We now define the mapping $S_t \to X_t$, where X_t is the vector sequence: $X_t = [\mathbf{x}_{t1}\mathbf{x}_{t2}\dots\mathbf{x}_{tL}]$. Let $\mathcal{A}$ be the symbol alphabet. $|\mathcal{A}|$ denotes the number of the elements in $\mathcal{A}$. Each symbol s_{ti}, $i = 1\dots L$ is then encoded into $|\mathcal{A}|$-dimensional vector $\mathbf{x}_{ti}$ so that all elements in $\mathbf{x}_{ti}$ are zeros except one corresponding to the index of the symbol s_{ti} in $\mathcal{A}$.

For example, if $\mathcal{A} = \{A, B, \dots, Z\}$ and $|\mathcal{A}| = 26$, the symbols in the alphabet are converted into vectors in the following way: $A \to [100\dots0]$, $B \to [010\dots0]$, $\dots$, $Z \to [000\dots1]$ and the converted symbol string is the sequence of the corresponding vectors. E.g., the string "BAD" would be converted to: $[0100\dots0]^T[1000\dots0]^T[0001\dots0]^T$.

4.2 Distance computation

DTW addresses to the problem of finding an optimal alignment between the encoded strings. This can be computed by dynamic programming [13, 14]. Let X_t and M_k denote two vector-encoded strings and F the warping function which performs the alignment. F is a sequence of index pairs which defines the mapping between the elements i of the sequence X_t and elements j of sequence M_k:

$$F = [z(1), z(2), \dots, z(p), \dots, z(P)], \quad (3)$$

where $z(p) = [i(p), j(p)]$ and P is the number of the points in the alignment. DTW-based distance is computed as a sum of distances between the sequence elements along the alignment:

$$D(X_t, M_k) = \sum_{p=1}^{P} d[z(p)], \quad (4)$$

where $d[z(p)]$ is the distance between the vectors of the sequences M_k and X_t indexed by $z(p)$. Euclidean distance $||X_t(i(p)) - M_k(j(p))||$ can be used. The cumulative distance in Eq. (4) can be divided by P or the sum of the sequence lengths.

The warping function can be computed in the two-dimensional trellis. Let $g(i,j)$ denote the path variable which gives the optimal warping. The trellis is first initialized:

$$\begin{aligned} g(0,j) &= \min \begin{cases} 0 & j = 0 \\ \infty & j > 0 \end{cases} \\ g(i,0) &= \min \begin{cases} 0 & i = 0 \\ \infty & i > 0 \end{cases} \end{aligned} \quad (5)$$

Dynamic programming follows:

$$g(i,j) = \min \begin{cases} g(i, j-1) + d(i,j) \\ g(i-1, j-1) + d(i,j) \\ g(i-1, j) + d(i,j), \end{cases} \quad (6)$$

$d(i,j)$ is the distance between $X_t(i)$ and $M_k(j)$. The index i goes from 1 to L_{X_t} and j from 1 to L_{M_k}, L_{X_t} and L_{M_k} denoting the lengths of X_t and M_k, respectively. In accordance with Eq. (4) we can now define:

$$D(X_t, M_k) = g(L_{X_t}, L_{M_k}). \quad (7)$$

From $g(i,j)$ we can obtain the warping function $F = z(p)_{p=1}^{P}$ which minimizes the sum of distances in Eq. (4). As an initialization we set:

$$z(P) = (L_A, L_B). \quad (8)$$

The rest of the warping path is obtained by backtracing the trellis:

$$z(p-1) = \arg\min \begin{cases} g(i, j-1) \\ g(i-1, j-1), \\ g(i-1, j) \end{cases} \quad (9)$$

where index p goes from $P-1$ to 1. Because of the initialization of $g(i,0)$ and $g(0,j)$, $z(1)$ will be $(1,1)$.

There are several modifications to the basic scheme described here which can be implemented, e.g., various slope constraints and weightings can be added to the warping function [13].

In case of symbols instead of feature vectors as the elements of the sequences, the corresponding distance is called Levenshtein or edit distance [11, 14]. The purpose of using vectors in this work is that they allow smooth symbol interpolation. This is explained in the following.

4.3 Model adaptation

Let us first investigate two vectors $\mathbf{m} = M_k(j(p))$ and $\mathbf{x} = X_t(i(p))$ from sequences M_k and X_t. The average of two vectors is simple to compute and we can use the model update formula of the original SOM algorithm:

$$\mathbf{m}' = \mathbf{m} + h(c,k,t)(\mathbf{x} - \mathbf{m}). \quad (10)$$

The entire sequences M_k and X_t are averaged by computing the sequence of vector averages along the alignment $F = z(p)_{p=1}^{P}$ of M_k and X_t .

Let $l(p)$ denote the "position" of the averaged sequence element. It is defined as the weighted average of the two indexes in $z(p) = [i(p), j(p)]$:

$$l(p) = (1 - h)j(p) + hi(p), \tag{11}$$

where h stands for the shorthand notation of $h(c, k, t)$. It is desirable that the length of the average sequence is proportional to L_{M_k} and L_{X_t}, the lengths of the sequences M_k and X_t. Let us denote the updated model M_k as M'_k. We can determine its length being:

$$L_{M'_k} = \lfloor (1 - h)L_{M_k} + hL_{X_t} + 0.5 \rfloor, \tag{12}$$

i.e., the length of the updated model sequence is a weighted average of its old length and the length of the current input rounded to the nearest integer number. A reasonable sampling interval from $z(p)$ is that $l(n+1) - l(n)$ is one, and index n goes 1 to $L_{M'} - 1$. The first value of l is set $l(1) = 1$.

If $l(n)$ does not coincide with any single point of $z(p)$ it can be interpolated between two nearest points in the warping path. Let us denote the indexes of these points by p_1 and p_2. They must satisfy:

$$l(p_1) < l(n) < l(p_2) \tag{13}$$

The nth element of the new model sequence can then be interpolated between two vector averages $\mathbf{m}_1$ and $\mathbf{m}_2$ corresponding to the warping points $z(p_1)$ and $z(p_2)$:

$$\mathbf{m}'(n) = \frac{q_1\mathbf{m}_1 + q_2\mathbf{m}_2}{q_1 + q_2}, \tag{14}$$

linear interpolation weights q_1 and q_2 are determined as $q_1 = l(p_2) - l(n)$ and $q_2 = l(n) - l(p_1)$, and $\mathbf{m}_1$ and $\mathbf{m}_2$ are computed according to Eq. (10).

4.3.1 Note on the learning rate

In the model adaptation, there are two averages to be made. One is the average between the vectors and another is for the sequence lengths. The feature vectors of the sequences will get adapted even when small learning rate values h are used. But the length of the updated model sequence is rounded to the nearest integer according to Eq. (12). The length adaptation does not have any effect if the learning rate is too small. Therefore, in order to get representative model sequences on the map, the learning rate should be sufficiently large in the beginning of the training, especially if the random model initialization is used.

This concludes the SOM training for variable-length sequences. The model sequence updating is performed as a weighted average of the input sequence and the model sequence according to Eq. (2) computing the vector averages along the warping path.

5 Examples

The first example shows the capability of the SOM for interpolating the data. The following five strings were in the dataset: "CAT", "CATTLE", "BAT", "BATTLE", and "BATTLEFIELD". Hexagonal SOM with 92 map units was used. The models were initialized by 26-dimensional random vectors, values were between one and zero. Each vector element corresponded to one symbol in alphabet A...Z. The initial models corresponded to the symbol sequences with only one symbol position in them. The map was trained in two stages with 1000 training cycles in each. The Gaussian neighborhood function was used in Eq. (2):

$$h(c(t), i) = \alpha(t) \exp[0.5d(c(t), i)^2/\sigma^2], \tag{15}$$

where $d(c(t), i)$ is the Euclidean distance between the coordinates of the nodes $c(t)$ and i on the map grid. In the first 1000 training cycles the effective width of the Gaussian neighborhood function σ decreased linearly from 10 to 3, and α decreased linearly from 0.5 to 0.01. In the second stage the neighborhood width was kept fixed, σ was 3, and another 1000 training cycles were performed. The resulting SOM is shown in Fig. 2. Because the lengths of the model sequences are adapted in addition to the vector elements, the models on the trained SOM correspond to the data strings despite their initialization.

Soft symbol vector sequences can be quantized into discrete symbol strings by extracting the symbol in each position which has the largest vector element. This is shown in Fig. 2(b).

Several larger SOMs were also constructed using a 100.000 word dictionary as the input data, one example is shown in Fig. 3. Random initialization was used and no problems were encountered during the training process.

6 Discussion

In case of discrete symbol strings, the Levenshtein distance computes the number of symbol changes between two strings. Some applications favor the use of weighted Levenshtein distance, where different symbol substitutions, deletions and insertions may have different weightings [17]. Similar approach can be embedded also in the vectorial representations of the symbols. Instead of the plain Euclidean distance, we can compute the weighted distance between two symbol representing vectors $\mathbf{a}$ and $\mathbf{b}$:

$$d(\mathbf{a}, \mathbf{b}) = [\mathbf{a} - \mathbf{b}]^T W[\mathbf{a} - \mathbf{b}], \tag{16}$$

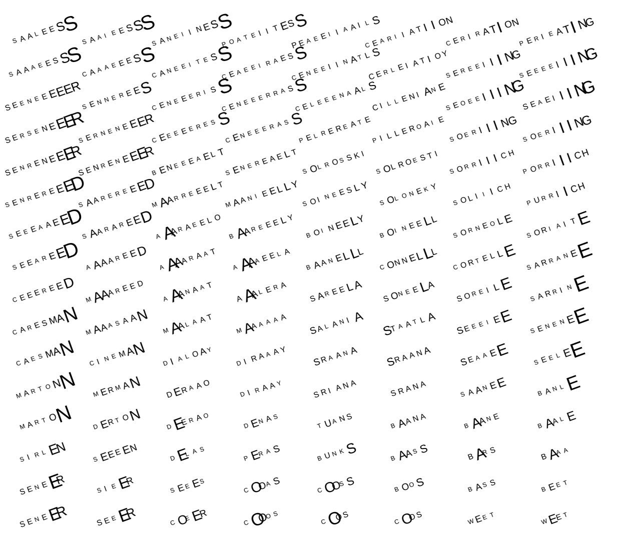


Figure 3: *128-unit SOM has been trained with the 100.000-word English dictionary as input data. Vector sequences of the trained models have been converted into discrete symbols by selecting the symbol with the largest value in each vector position. Font size represents the dominance of the symbol in each vector.*

where W denotes the weighting matrix.

The soft symbol representation used in the present work allows probabilistic interpretation of the data. For visualization purposes, this can be utilized e.g. by choosing the font size of the symbol according to its value in the vector as shown in Fig. 2(b) and 3.

7 Conclusions

This work presented an online algorithm for the SOM with symbol strings as data. Symbol strings were converted into vector sequences which enabled the computation of smooth averages. Dynamic time warping was used for comparing the input strings against the model sequences and arithmetic averaging was used for updating the models.

The main advantage of the proposed method over the previous algorithms for constructing the SOM for symbol strings is the smooth interpolation ability due to the vectorial representation of the symbols. The soft symbol representation enables also probabilistic interpretation of the data.

Acknowledgements

This work was supported by Academy of Finland, project no. 44886 "New information processing principles" (Finnish Centre of Excellence Programme 2000-2005).

References

[1] S. Aaltonen and K. Torkkola, "Merkkijonojen itseorganisaatio", *Technical report Tik-61.198* (in Finnish), Helsinki University of Technology, 1989.

[2] M. Andrade, G. Casari, C. Sander, and A. Valencia, "Classification of protein families and detection of the determinant residues with an improved self-organizing map", *Biological Cybernetics*, vol. 76, pp. 441-450, 1997.

[3] I. Fischer and A. Zell, "String Averages and Self-Organizing Maps for Strings", *Proceedings of the Second ICSC Symposium on Neural Computation (NC'2000)*, pp. 208-215, 2000.

[4] T. Kohonen,"Self-Organized formation of topologically correct feature maps", *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[5] T. Kohonen,"Median strings", *Pattern Recognition Letters*, vol. 3, pp. 309-313, 1985.

[6] T. Kohonen, *Self-Organizing Maps*, Springer, 1995.

[7] T. Kohonen, "Self-Organizing Maps of Symbol Strings", *Technical Report A 42*, Helsinki University of Technology, Laboratory of Computer and Information Science, 1996.

[8] T. Kohonen and P. Somervuo "Self-Organizing Maps of Symbol Strings With Application to Speech Recognition", *Proceedings of the Workshop on Self-Organizing Maps (WSOM'97)*, pp. 2-7, 1997.

[9] T. Kohonen and P. Somervuo "Self-Organizing Maps of Symbol Strings", *Neurocomputing*, 21, pp. 19-30, 1998.

[10] T. Kohonen and P. Somervuo "How to make large self-organizing maps for nonvectorial data", *Neural Networks*, vol. 15, no. 8-9, pp. 945-952, 2002.

[11] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals", *Cybernetics and Control Theory*, vol. 10, no. 8, pp. 707-710, 1966.

[12] L. Rabiner and J. Wilpon, "Considerations in applying clustering techniques to speaker-independent word recognition", *J. Acoust. Soc. Am.*, vol. 66, no. 3, pp. 663-672, 1979.

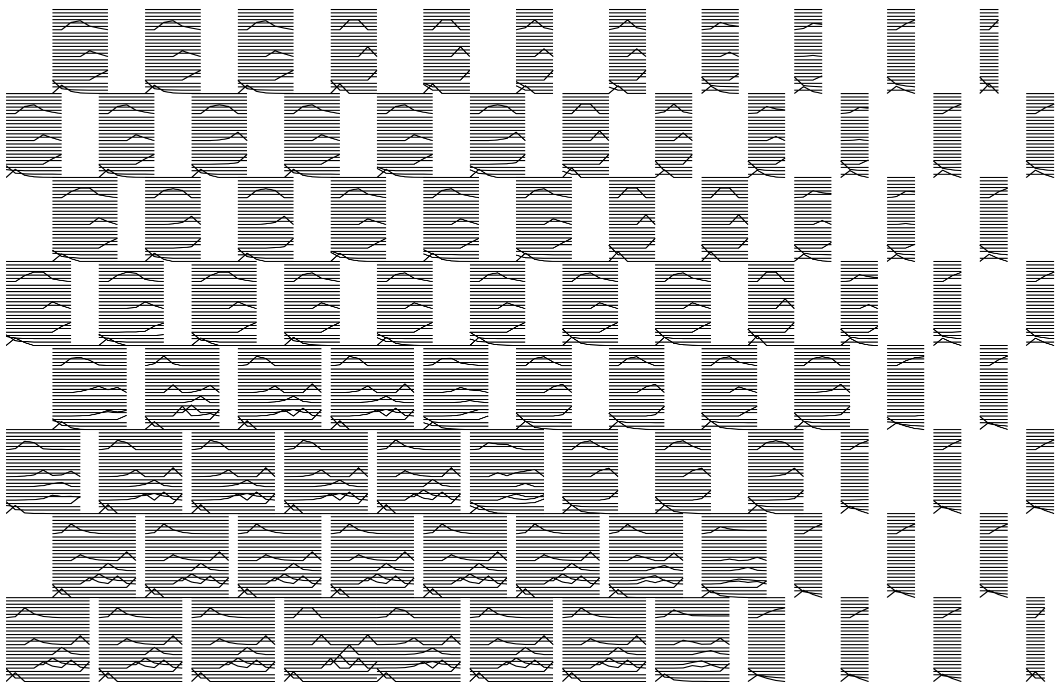
[13] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition", *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 26, no. 1, pp. 43-49, 1978.

[14] D. Sankoff and J. Kruskal, *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*, Addison-Wesley, 1983.

[15] P. Somervuo and T. Kohonen, "Self-organizing maps and learning vector quantization for feature sequences", *Neural Processing Letters*, vol. 10, no. 2, pp. 151-159, 1999.

[16] P. Somervuo and T. Kohonen, "Clustering and Visualization of Large Protein Sequence Databases by Means of an Extension of the Self-Organizing Map", *Proceedings of the Discovery Science (DS'2000), Lecture Notes in Artificial Intelligence 1967*, pp. 76-85, 2000.

[17] E. Tanaka and T. Kasai, "Synchronization and substitution error-correcting codes for the Levenshtein metric", *IEEE Trans. Information Theory*, vol. 22, no. 2, pp. 156-162, 1976.


(a) Soft symbol sequence models

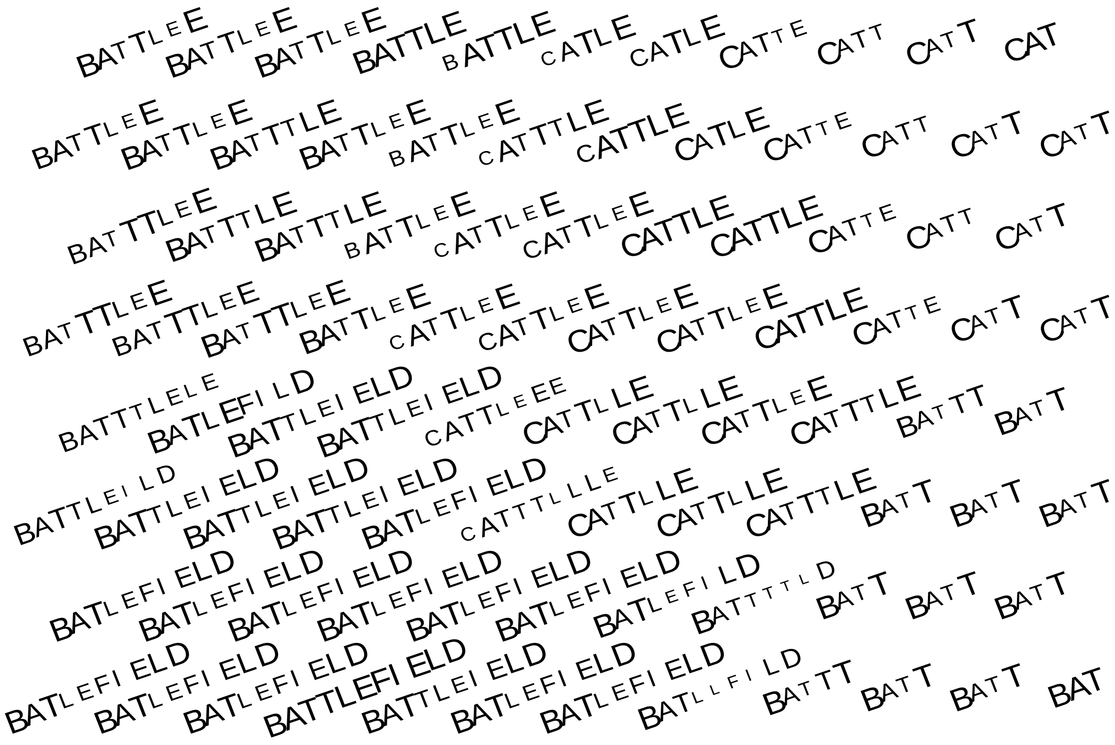

BATTLEE BATTLEE BATTLEE BATTLE BATTLE CATLE CATLE CATTE CATT CATT CAT
BATTLEE BATTLEE BATTTLE BATTLEE BATTLEE CATTTLE CATTLE CATLE CATTE CATT CATT CATT
BATTTLEE BATTTLE BATTTLE BATTLEE CATTLEE CATTLEE CATTLE CATTLE CATTE CATT CATT
BATTTLEE BATTTLEE BATTTLEE BATTLEE CATTLEE CATTLEE CATTLEE CATTLEE CATTLE CATTE CATT CATT
BATTTLELE BATLEFILD BATTLEIELD BATTLEIELD CATTLEEE CATTLLE CATTLLE CATTLEE CATTTLE BATTT BATT
BATTLEILD BATTLEIELD BATTLEIELD BATTLEIELD BATLEFIELD CATTTLLLE CATTLLE CATTLLE CATTTLE BATT BATT BATT
BATLEFIELD BATLEFIELD BATLEFIELD BATLEFIELD BATLEFIELD BATLEFIELD BATLEFILD BATTTTTLD BATT BATT BATT
BATLEFIELD BATLEFIELD BATLEFIELD BATTLEFIELD BATTLEIELD BATLEFIELD BATLEFIELD BATLLFILD BATTT BATT BATT BAT
(b) Quantized symbol sequence models

Figure 2: *92-unit hexagonal SOM for symbol sequences. Input data consisted of five strings: "CAT", "CATTLE", "BAT", "BATTLE", and "BATTLEFIELD". Fig. (a) contains 26-dimensional vector sequences, each vector element corresponding to one symbol in the alphabet A...Z. These sequences have been converted into discrete symbol strings in Fig. (b) by selecting the symbol with the largest value in each vector. The size of the font indicates the value of the symbol in the vector representation; large size corresponds to large value.*