Episodic Memory in Minicolumn Associative Knowledge Graphs

Basawaraj, Janusz A. Starzyk, *Senior Member, IEEE*, and Adrian Horzyk, *Senior Member, IEEE*

***Abstract*— A generalization of active neural associative knowledge graphs (ANAKGs) to their minicolumn form is presented in this paper. Each minicolumn represents a single symbol, and the activation of an individual neuron in a minicolumn depends on the context of the activation of the presynaptic neuron. The implemented memory model combines the ANAKG associative spiking neuron idea with the idea of the hierarchical temporal memory. This new associative memory organization preserves all properties of ANAKG memories, such as storage of knowledge based on the association of spatiotemporal input sequences, self-organization, quick learning, and recall of the sequential memories, while increasing the recall quality and the memory capacity. The recall quality advantage of the new approach over ANAKG increases with the length of the recalled episodes and the number of neurons used in each minicolumn. We introduced a new distance measure to compare the recalled sequences and defined a recall quality to determine the memory capacity. Performed tests confirmed our claims. Additional tests were performed to illustrate the computational complexity and the efficiency of the developed approach.**

***Index Terms*— Associative episodic memory, distance measure, knowledge representation, minicolumn structure, recall quality.**

I. Introduction

MEMORY plays an important role in a cognitive system, providing it with the knowledge about its environment and how to deal with it. Its structure self-organizes as a result of the past observations, actions, and their consequences [1]. The learning process includes changes in the long-term memory cells and the synaptic connections between neurons. Associations between neurons reflect contexts for the learning and representation building process [2].

Memory is composed of distinct systems that can be divided into declarative (explicit or conscious) and nondeclarative (implicit or subconscious). Declarative memories are further divided into semantic and episodic memories [3]. While semantic memory is a structured record of facts, concepts, and knowledge about the world acquired over the lifetime, episodic memory is a representation of personal experiences and specific events (time, place, emotions, and other contextual knowledge) that can be explicitly stated. Episodic memory supports learning in the semantic memory by providing a recollection of past events. Both types of memory require the storage of sequential information.

Associative networks are content addressable and are able to retrieve stored data based on only a part of what was stored [4]. They are resistant to noise and can detect missing data and sensory failures [5]. Models of associative networks with feedback loops, called recurrent neural networks (RNNs), can be trained to predict the next output symbol after reading a stream of input symbols. In [6], gradient-based RNNs were used to retrieve the memories of the stored input sequences. In [7], an unsupervised algorithm that, using RNNs, learns fixed-length feature representations of sentences, paragraphs, and documents was proposed. The Penn Corpus and Switchboard, with about 1 million and 4 million words, respectively, were used in [7]. Similar to RNNs, this algorithm is trained to predict words in a document given an input context. Memory networks [8] are a new class of learning models that combine the input content with the dynamic knowledge base stored in the long-term memory to predict the output. Memory networks represent the input information in the form of features and are capable of generalization to produce the desired response.

In response to demand for services based on speech recognition and large knowledge bases like Wikipedia, researchers in recent years have focused on contextual question answering (QA). Direct approaches to QA such as string matching are ineffective [9], and solutions that include recursive neural networks such as QA neural network with trans-sentential averaging [9] are becoming popular. Neural Turing machines (NTMs) combine the concept of neural network learning and classical Turing machines to retrieve context-based input information [10]. NTMs can learn simple algorithms from input and output examples and use them to generalize. Compared with RNN long short-term memory (LSTM) [6], NTMs show better accuracy over longer sequences in recall and copy tasks [10].

Semantic knowledge and short-term memory must cooperate to provide a context-based scene understanding and recall of the useful operations that the system performed in an open environment. Semantic memory aggregates representation of the training data and forms a context-searchable knowledge base. This memory is obtained by binding the semantic contexts for all trained objects and linking their neuronal representations together. Semantic memory can be built using an active neural associative knowledge graph (ANAKG) that uses

Manuscript received August 24, 2018; revised April 2, 2019; accepted June 29, 2019. Date of publication August 5, 2019; date of current version October 29, 2019. This work was supported by the National Science Centre of Poland under Grant DEC2016/21/B/ST7/02220 and Contract AGH 11.11.120.612. *(Corresponding author: Adrian Horzyk.)*

Basawaraj is with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701 USA.

J. A. Starzyk is with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701 USA, and also with the Department of Applied Information Systems, University of Information Technology and Management in Rzeszów, 35-225 Rzeszów, Poland.

A. Horzyk is with the Department of Automatics Biocybernetics and Biomedical Engineering, AGH University of Science and Technology, 30-059 Kraków, Poland (e-mail: horzyk@agh.edu.pl).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2927106

the associative spiking neuron (as-neuron) model presented in [11].

RNNs that use artificial minicolumns have a much larger storage capacity than ordinary networks in which each neuron represents a single concept as discussed in [12]. Minicolumn refers to one or more neurons that function as a unit. We use this property of artificial minicolumns to increase the memory capacity and improve the quality of knowledge representation in ANAKG memories.

In this paper, we present a generalization of ANAKGs to their minicolumn form known as lumped minicolumn associative knowledge graph (LUMAKG) memory. We present LUMAKG organization and a learning process to establish spatiotemporal associative connections between neurons. In LUMAKG, each symbol is represented several times following the idea of minicolumn organization presented in [12]. LUMAKG uses the same spiking neuron model as ANAKG and similar self-organization principles. The most significant difference between the two memory structures is that LUMAKG uses columnar organization and a new mechanism for selection of synaptic connections between neurons. While the columnar organization increases the memory capacity, the new mechanism for synaptic connections improves the resolution of context-based sequence recognition.

The major contribution of this work is to demonstrate that LUMAKG organization preserves all the properties of ANAKG memories such as storage of knowledge based on the association of spatiotemporal input sequences, self-organization, quick learning, and recall of the sequential memories, while increasing the recall quality and the memory capacity. LUMAKG is better suited to store and recall time domain sequences that are critical to formulating a context-based episodic memory.

The remainder of this paper is organized as follows. Section II briefly introduces the major features of ANAKG memories. Section III contains the organizing principles of LUMAKG memory and a description of the algorithm to locate neurons with predicted activation (PA). Section IV presents the LUMAKG design example and details of the memory organization algorithm. Section V describes a testing methodology and introduces the reciprocal word position (RWP) distance and recall quality used in the evaluation of the memory properties. Section VI discusses the conclusions.

II. Semantic Memories

An ANAKG can be used to build semantic memories. The knowledge graphs are dynamically obtained by adding associative neurons and changing their synaptic connections based on the input sequences and activation levels of presynaptic and postsynaptic neurons. If the updated synaptic weights provide incorrect activations of postsynaptic neurons, then the previously activated neurons create inhibitive connections to the incorrectly activated neurons. The gradual activation and relaxation of ANAKG neurons enable them to represent sequences of elements (objects) in their previous contexts.

ANAKG networks are built from as-neurons, with receptors sensitive to input stimuli and effectors transforming neuronal stimuli into output data [11], [13], [14]. The as-neurons can be very quickly adapted to represent any given set of training sequences of elements in a neural graph structure that integrates and associates them. The ANAKG networks demand only a single presentation of each training sequence to create a neuronal structure and compute all weights. This approach yields the neural graph structure much faster than training routines of artificial neural networks. It is also easier to compute than spiking neural networks, which require solving several differential equations. The adaptation process of the ANAKG uses the so-called *synaptic efficacy* (1) computed according to the time that elapsed between activities of the presynaptic and postsynaptic as-neurons that were activated in close time succession. The shorter this period is, the bigger the impact on the synaptic efficacy is. The synaptic efficacy is also dependent on the frequency of synaptic stimulation to the postsynaptic neuron. It significantly simplifies the adaptation process in comparison to other models of neurons, both spiking and those that use nonlinear activation functions.

Assume that we have a training sequence set $\mathbb{S} = \{S^1, \ldots, S^N\}$ consisting of sequences $S^n = [E_1^n, \ldots, E_m^n, \ldots, E_{m+r}^n, \ldots, E_{K_n}^n]$, of possibly different lengths, where E_m^n is an element (object). The synaptic efficacy is computed for each pair of connected as-neurons N_m and N_{m+r} representing two elements E_m and E_{m+r} in each training sequence S^n that contains them. Elements E_m and E_{m+r}, represented by as-neurons N_m and N_{m+r}, in a subset of all training sequences are separated in time by a number of other elements $(r-1)$, where $r \geq 1$. The time difference between observation of E_m and E_{m+r} elements stimulating neurons N_m and N_{m+r} affects the computation of various components of the sum (1) that defines synaptic efficacy. The final synaptic efficacy for this synapse takes into account all training sequences that contain time ordered succession of elements E_m and E_{m+r}. The synaptic connection between as-neurons representing elements E_m and E_{m+r} is denoted here as $N_m \rightarrow N_{m+r}$

$$\delta_{E_m, E_{m+r}} = \sum\nolimits_{\{(E_m, E_{m+r}) \in S^n \in \mathbb{S}\}} \left(\frac{1}{1 + \frac{\Delta t^A - \Delta t^C}{\theta_{N_{m+r}} \cdot \Delta t^R}} \right)^{\tau} \quad (1)$$

where

- δ synaptic efficiency;
- Δt^A period of time that lapsed between stimulation of synapse between N_m and N_{m+r} neurons and activation of the postsynaptic neuron N_{m+r} during training;
- Δt^C period of time necessary to charge and activate a postsynaptic neuron N_{m+r} after stimulating synapse between N_m and N_{m+r} neurons (here $\Delta t^C = 20$ ms);
- Δt^R maximum period of time during which postsynaptic neuron N_{m+r} recovers and returns to its resting state after its charging that was not strong enough to activate this neuron (here $\Delta t^R = 300$ ms);
- $\theta_{N_{m+r}}$ activation threshold of the postsynaptic neuron N_{m+r} (here $\theta_{N_{m+r}} = 1$);
- τ context influence factor changing the influence of the previously activated and connected neurons on the postsynaptic neuron N_{m+r} (here $\tau = 4$).

Synaptic efficacy is a measure of how strong a given input stimulation of the synapse influences the postsynaptic neuron activity due to the elapsed time between activations of presynaptic and postsynaptic as-neurons. It weighs and sums up all related activities of the connected neurons during such an adaptation process.

ANAKG consolidates representations of many training sequences in such a way that repeated elements occurring in various training sequences are represented by single as-neurons. Such neurons bind training sequences together and thanks to contextual connections between them it is possible to retrieve many of these sequences using unique initial contexts for recalling them [11], [13], [14]. When new or nonunique contexts are used, the ANAKG retrieves the most frequent training sequences or new sequences that are built either from parts of the training sequences or their associated elements.

The synaptic efficacy is used to compute a *synaptic permeability,* which is used in this model as a *connection weight* (2). The synaptic permeability values of each synapse are between 0 and θ, where θ is a threshold value of the postsynaptic neuron N_{m+r}. The range of permeability values emphasizes the influence of the presynaptic neuron N_m on the postsynaptic neuron N_{m+r}. It is computed after the activity of the presynaptic neuron N_m by considering synaptic efficacy that contains its influence on the postsynaptic neuron activity using the following equation [13]:

$$w = \theta \frac{\eta\delta}{\eta\delta + \eta^2 - \delta^2} \tag{2}$$

where

- w synaptic permeability, that is, connection weight;
- η number of activations of a presynaptic neuron N_m during training for training sequence set $\mathbb{S}$;
- δ synaptic efficacy computed for this synapse.

ANAKG-based semantic memory can associate distant time events and trigger the recalling processes automatically, taking into account the given context and semantic relations between objects represented in the neuronal graph structure. Semantic relations are automatically created on the basis of real relationships between objects presented to this memory in the form of sequential patterns. They are weighted according to the strength or the frequency of represented relations in their wider context comprising previous events. Such a strategy makes it possible to represent various concepts from different points of view and in different contexts, forming knowledge about them. Finally, the ANAKG memory can generalize knowledge gained during the adaptation process based on the presented training data. It can generate new responses that were not previously observed but reflect knowledge accumulated by the system and the context of the current events [15]. The generalization is a result of the association and aggregations of data, symbols, features, objects, and subsequences that occur in the training data.

ANAKG networks are robust to distortions in the input signal, and to some degree their effectiveness resembles that of the LSTMs [6] that can store short-term sequential information over longer periods of time through its gating system. LSTMs are a type of RNNs that are capable of learning long-term dependencies. While LSTMs and RNNs have a similar form, a recurrent hidden layer consisting of a chain of recurrently connected neural network modules, the neural network modules themselves are different. The neural network modules in RNNs have a very simple structure, for example, a single memory cell, whereas the neural network modules in LSTM have multiple neural network layers, for example, one or more memory cells and gates to control the flow of information. In contrast to LSTMs, ANAKG networks are easy to train and do not require supervised learning, which makes them a better choice for natural learning in an open environment.

A. Sample ANAKG Structure

This section describes the creation of a sample ANAKG structure for the following training sequence set: *I have a monkey. My monkey is very small. It is very lovely. It likes to sit on my head.* The developed ANAKG structure is shown in Fig. 1. Here, each as-neuron represents a single word, and the numbers under their names represent their number of activations (η) during the training phase. These numbers are also equal to the number of occurrences of each word in all training sequences. The small circles represent the postsynaptic elements of the synapses, and a red dot inside them means that the value of the synaptic weight is equal to the threshold of the postsynaptic neuron (θ), that is, the activation of this synapse is sufficient to activate the postsynaptic neuron, whereas numbers in the postsynaptic elements (small circles) represent synaptic permeability values (w) as a percentage of the threshold value (θ). Similarly, the crescent shape denotes the presynaptic elements and shows the direction from which the stimuli come.

Fig. 1(A) shows the neural network following the presentation of the first sentence: *I have a monkey.* Following the process described in [11], for each word in the sentence, we first check if there exists an as-neuron that reacts to the presented word. If none of the existing as-neurons are activated, a new as-neuron is created. This process is repeated for all words in this sentence. If a word is repeated, the same neuron represents it. The efficiencies of synaptic connections following the stimulation of presynaptic as-neurons were computed using (1). The connection weights were computed according to (2). Because this is the first training sentence, and there are no repetitions of words, the activation of any as-neuron is sufficient to activate the postsynaptic as-neuron representing the next word in the sentence. Besides, as-neurons representing subsequent words in the sequence are also stimulated by the as-neurons representing previous words of the trained sequence, establishing the context of the following words. In Fig. 1(A), this context represented by the additional connections (small circles with numbers) does not influence the result of stimulation significantly, but these connections play a substantial role when next sentences partially composed from the same words will be represented by this structure [Fig. 1(B)–(D)].

Fig. 1(B) shows the ANAKG network after the presentation of the second sentence: *my monkey is very small.* Four new as-neurons representing the new words {*my, is, very, small*} are

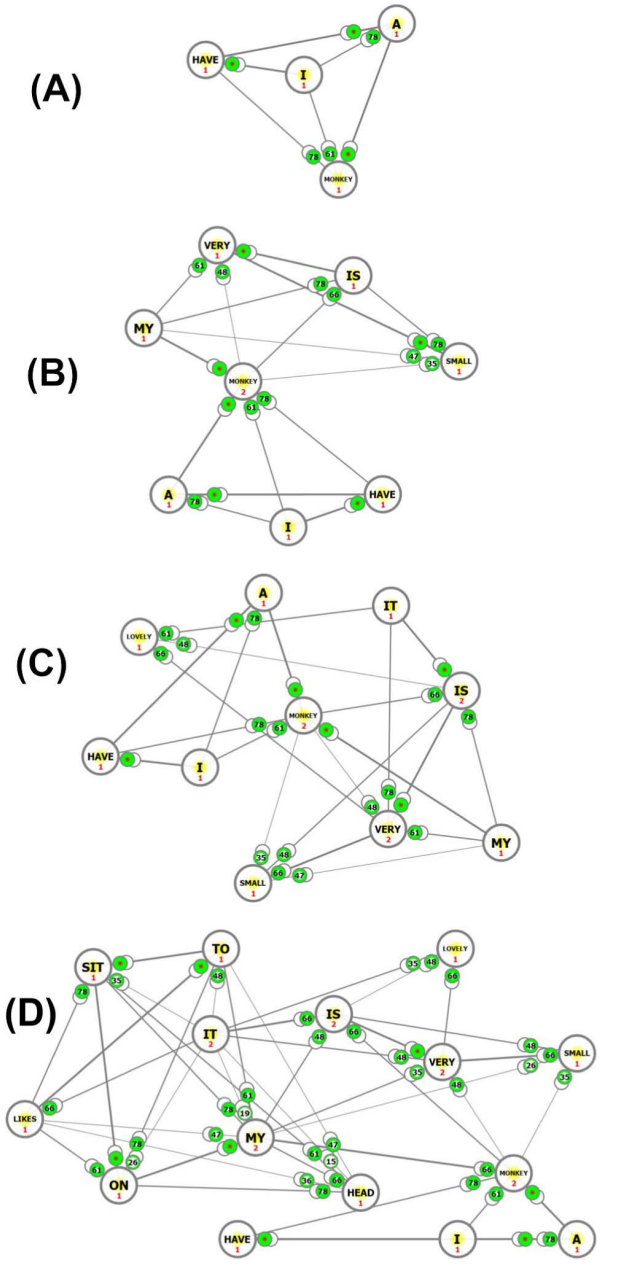

Fig. 1. First four steps of a sample ANAKG structure developed after adding the following sequences: (A) I have a monkey, (B) My monkey is very small, (C) It is very lovely, and (D) It likes to sit on my head, according to the associative process using (1) and (2).

created, and the synaptic connections to all their predecessors in the sentence are added. The synaptic efficiencies and connection weights are calculated according to (1) and (2), respectively. Note the aggregation of the representation of the word *monkey* occurring in both of the currently trained sentences. When a word is shared between a few training sentences, the neuron representing this word does not stimulate neurons representing subsequent words sufficiently to activate them. For instance, the word *monkey* cannot activate the neuron *is* from the second sentence by itself, but stimulation of the neuron representing even earlier word *my* (the context) is required. Similarly, following the training with the third sentence: *it is very lovely*, the synaptic efficiency and connection weights between *very* and *small* change due to the occurrence of *lovely* following the word *very*. The activation of the presynaptic as-neuron *very* is not sufficient to activate the postsynaptic neuron *small* anymore [see Fig. 1(C)]. It is necessary to use the context of previously activated neurons {*my, monkey, is}* or {*it, is*} to adequately stimulate the neurons*small* and*lovely* to activate the right one according to its context. Fig. 1(D) presents the ANAKG network after the fourth training sentence (*It likes to sit on my head*) was added. Further descriptions of the ANAKG structure associative processes can be found in [11], [16], and [17].

III. Organization of LUMAKG

A. Minicolumn Organization of the Associative Memory

Hawking *et al.* [12] proposed a columnar organization of the associative memory and introduced cortical learning algorithms in which minicolumns were used to store sequential information in structures known as hierarchical temporal memory (HTM). HTM is a theory of intelligence based on neuroscience research, and structurally it is inspired by the multi-layer hierarchical network structure of the human neocortex. Since then, HTMs were further developed, and their properties were analyzed and tested. In [18], it was shown that HTM was able to continuously learn a large number of temporal sequences using an unsupervised learning neural network model. HTM was shown to have similar accuracy as another state-of-the-art sequence learning algorithms such as echo state networks [19] or LSTM [18]. However, they also show some drawbacks like larger sensitivity to temporal noise than LSTM [18]. ANAKG memories do not have this drawback of HTM networks because as-neurons use the time delay of the input signal to make associations, and ANAKG associates not only the individual inputs but also their sequences spread over time, greatly minimizing the effect of temporal noise (or any single event). The gradual change in sensitivity, activation threshold, synaptic weights, and connections to other neurons and sensors that results from the model parameters also helps to minimize the effect of temporal noise. Thus, improving ANAKG by introducing a minicolumn structure to its architecture provides a better associative memory capable of storing spatiotemporal relations between data. Continuous learning from input data and rapid adaptation to changing environmental conditions are desired properties of machine learning [18], so it is important that the algorithm can recognize and learn new patterns quickly. ANAKG memories have this property.

Both in HTM and ANAKG, neurons do not perform a simple weighted sum of their inputs as in most neural network models [20]–[22], but integrate them over time. This is similar to spiking neuron networks [23]. Spiking neurons are biologically motivated and produce patterns similar to biological neurons. Several computationally efficient models of spiking neurons have been developed [24]. Networks of spiking neurons spontaneously self-organize into groups and generate polychronic patterns of activity, and this property is believed to be necessary for cognitive neural computations, symbol grounding, attention, and consciousness [25]. ANAKG

achieves similar properties to spiking neurons by using a much simpler spiking neuron model and self-organization principles to capture the spatiotemporal relationship between data [13]. LUMAKG maintains these properties of ANAKG while increasing its recall quality, memory capacity, and resolution.

Following HTM organization, we replace each neuron in ANAKG with a minicolumn of several neurons, where all the minicolumn neurons represent one unique symbol (e.g., a single word). Individual neurons in the active columns represent information regarding the learned temporal context, and they may be activated by different learned temporal contexts. Like in HTM, the neurons in LUMAKG receive three types of inputs. The input from the lower layer network carries the sensory information and is used to recognize the learned sequences, the input from the higher layer represents feedback prediction, and the input from the same layer represents context-based prediction and lateral inhibition used to create self-organizing maps.

While the neurons in a minicolumn are duplicates of each other, their inputs, outputs, and synaptic connections (weights) are not. Using the design principles from HTMs [12], the inputs and outputs are distributed across all the minicolumn neurons so that multiple sequences can be represented using the same set of minicolumns. Individual neurons in each minicolumn use the ANAKG algorithm to establish associative connections and their synaptic weights. Like ANAKG neurons, LUMAKG neurons modify their thresholds to stimulate learning by various minicolumns and their neurons.

Like in HTM, LUMAKG minicolumns have three output states, active from feed-forward input (can be input from the sensor), active from lateral input (representing a prediction), and inactive. Thus, LUMAKG neurons can fire even without sensory input stimulation. In the predictive mode, activation of neurons from the lateral input is used to complete the sequence. During learning of new sequences, prediction and input activation should match for the learning (changing the synaptic weights) to take place.

B. Organizing Principles of LUMAKG

We illustrate the design of LUMAKG memory by using sequences of words as its input. Each minicolumn in the developed structure represents a different word. Although a minicolumn-based memory is capable of handling raw sensory data to obtain its symbolic representations [26], we use this simplified approach where the input signals are the sequences of symbols. We do so in order to have a simple interpretation of the neurons' activities and to use simple measures to compare test results with ANAKG or other neural networks that use symbolic inputs. A distributed version of the minicolumn associative knowledge graph (DIMAKG) is currently under development.

The LUMAKG structure is obtained dynamically. New minicolumns and synaptic connections are added each time a new input sequence is provided to the network. Specifically, if a new symbol is observed, a new minicolumn is added, and at least one of its neurons is linked to other minicolumns establishing new synaptic connections. The organizing principles of LUMAKG are as follows.

1) Duplicate each symbol m times to form an individual symbol minicolumn.
2) If a neuron in a minicolumn is activated above its threshold from the associative connections, it is called to be in a *predictive mode*.
3) A sensory input activates either all the neurons in a given minicolumn that are in the predictive mode or the whole minicolumn if no neuron is in a predictive mode.
4) Activated neurons that were in a predictive mode are in PA.
5) An activated minicolumn without any neuron in a predictive mode has all the neurons in *unpredicted activation* (UA).
6) Synaptic weights of connections between activated neurons in the predecessor and the successor minicolumns are changed according to (2).

The number of neurons in each minicolumn m is set arbitrarily. In the NuPIC software that implements HTM memory, m is set to 32 [26], [27]. In the human cortex, the number of neurons in each minicolumn is between 80 and 120 (with twice this number in the visual cortex area) [28]. Since in our approach we demonstrate that the memory and its resolution depend on this number, m is an important network design parameter. We hypothesize that larger memory networks need a larger number of neurons in their minicolumns for their optimum performance.

C. LUMAKG Algorithm

According to the described organizing principles, the LUMAKG algorithm can be organized as follows. The individual steps of this algorithm are explained and illustrated using a design example.

The LUMAKG Algorithm:

I. Read the consecutive elements of the input sequence to activate the corresponding minicolumns.

1) Check if the symbol from the input sequence is represented by a minicolumn.
2) If it is not, add a new minicolumn.
3) Put all neurons of this new minicolumn in the state of UA.

II. Establish the predecessor–successor neurons in all the minicolumns activated by the input sequence.

1) Find the nonoverlapping sequences of the previously stored episodes.
2) Establish a sequence of linked PA neurons in all the activated minicolumns.

III. Update the synaptic weights in the synaptic connections between all predecessor-successor neurons.

Update all the synaptic weights between all the PA neurons in the predecessor and successor minicolumns according to the rules developed for ANAKG [11].

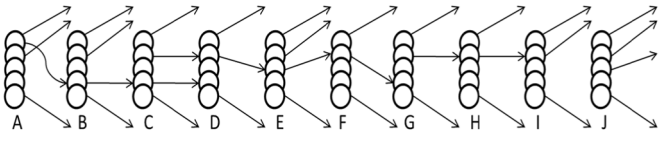

Fig. 2. Activated minicolumns with the existing synaptic connections.

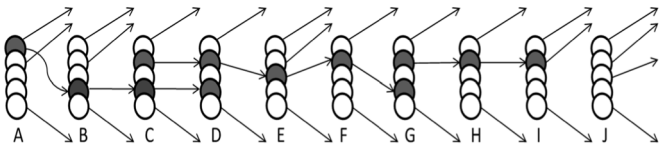

Fig. 3. LENs of the previously learned sequences of symbols.

IV. LUMAKG Design Example

Since the LUMAKG algorithm has a convoluted process for modification of synaptic connections, we use an example to illustrate how the algorithm works. First, we will illustrate how to find nonoverlapping sequences of the previously stored episodes in all the minicolumns activated by the input sequence (point II.1 of the LUMAKG algorithm). In this example, we assume for simplicity of the graphical illustration that the number of neurons in each minicolumn is equal to 5. This, however, should be optimized depending on the desired memory size.

For each consecutive activated minicolumn, activate all the neurons in the minicolumn that corresponds to the input symbol according to point C of the organizing principles. Typically, the first activated minicolumn has no PA neurons, unless it is considered in the broader context of associative learning and was a part of the previously stored episode. Thus, typically all neurons in the first activated minicolumn are in the state of an UA.

A. Finding Nonoverlapping Sequences

To better explain point II.1 of the LUMAKG algorithm, let us illustrate it with an example of a sequence of activated minicolumns. Let us assume that the sequence "A, B, C, D, E, F, G, H, I, J" was inputted to the LUMAKG memory and the corresponding minicolumns were activated as shown in Fig. 2. This sequence could represent a number of sentences with all different words like the following sentence: *I didn't really know how to cook these green plantains.*

If the previously obtained inputs contained sequences that used some of these words, then there will be synaptic connections between possibly different neurons in the corresponding minicolumns. For instance, suppose the previous inputs to LUMAKG memory contained the following sentences:

I didn't really know this. Nuns really know how to cook oysters. Don't cook these green mushrooms.

Then the corresponding synaptic connections could start at various locations in their minicolumns—as we can observe in Fig. 2.

In order to modify the existing synaptic connections or to introduce new connections while preserving the episodic storage, we need to find a sequence of activated neurons in these minicolumns that preserves the most significant episodes. This is accomplished by following the existing associative links to specific locations within each minicolumn.

We first identify which subsequences of the newly activated minicolumns were parts of the stored episodic memories. In Fig. 3, we show these neurons in the newly activated minicolumns that already have synaptic connections to other consecutive activated minicolumns and were parts of previously learned sequences. If activated, they will predict the activations of the corresponding postsynaptic neurons in the previously learned sequences. We call these neurons *linked episodic neurons* (LENs). In Fig. 3, we mark these LENs using a darker shade.

Following the directed links from each LEN, we can find related *predictive graphs of minicolumns* (PGMs). For instance, in Fig. 3, we can observe three PGMs: A–D, C–G, and G–I.

First, we find the PGM with the maximum number of minicolumns and declare all its linked neurons as PA neurons. Thus all linked neurons in PGM that contains minicolumns C–G are PA neurons. In this way, we take advantage of the previously stored episodic fragments, strengthening their joint probability of activations.

If a smaller PGM has minicolumns that overlap with the larger PGM, then its overlapping minicolumns are removed from the PGM. For instance, PGM composed of A–D has minicolumns C and D that are also a part of a larger PGM, and thus this PGM is reduced to two neurons A, B. If after reduction a PGM has less than two neurons, it is trivial and does not define any PA neurons. When PGMs overlap, we need means to determine where the new synaptic connections are added to represent the new contextual relationship observed. The process of removing overlapping minicolumns from smaller PGMs helps to find the nonoverlapping sequences of previously stored episodes and to determine where the new synaptic connections are added. Although we could find the nonoverlapping sequences of previous episodes by removing the overlapping minicolumns from the larger PGM, doing so can lead to fragmentation of stored episodes. Note that the removal of overlapping minicolumns from smaller PGM does not remove existing synaptic connections.

Similarly, the minicolumn G in the PGM graph G–I overlaps with the larger PGM graph C–G. Hence, this PGM graph (G–I) can only define two PA neurons in columns H and I. Such overlapping PGMs can be merged by removing the overlapping neurons from the smaller PGMs and adding new synaptic connections from the predecessor PGM to a successor PGM as illustrated in Fig. 4, where a predecessor PGM is the one that has minicolumns whose activation precedes activation of minicolumns in the successor PGM.

B. Establishing a Sequence of Linked PA Neurons

After merging of the overlapping PGMs, we may end up with more than one sequence of linked PA neurons (based on the linked episodes). Fig. 5 shows such a case in which the previously discussed sequence A–J is just a subsequence

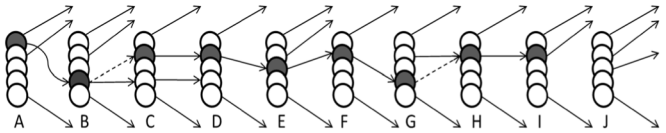

Fig. 4. Merging of three overlapping PGMs. A new synaptic connections are added from the end of the predecessor PGM (neurons A and B) to the first neuron of the successor PGM (neuron C), and from the end of the predecessor PGM (neurons C–G) to the first neuron of the successor PGM (neuron H).

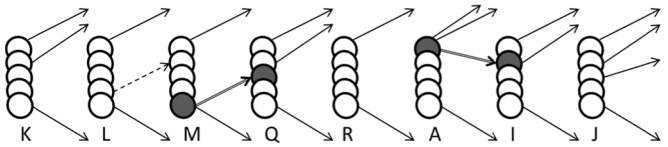

Fig. 5. Longer sequence of activated minicolumns.

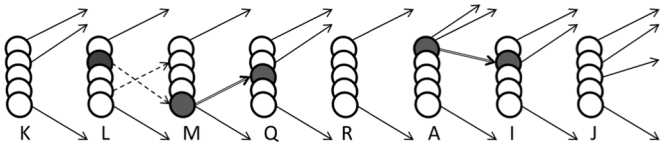

Fig. 6. New connection between the UA minicolumn L and a PA neuron in the minicolumn M.

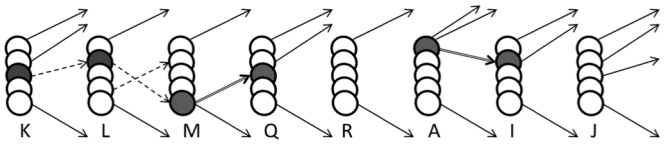

Fig. 7. New connection between UA minicolumn K and a PA neuron in minicolumn L.

that follows another sequence K–R. Here, for simplicity, we represent each sequence of the linked episodes by a single predecessor–successor link with a double solid line.

To establish a sequence of linked PA neurons in all the activated minicolumns that are needed in II.2 of the LUMAKG algorithm, we follow the steps specified in the locating PA neurons (LPANs) algorithm.

LPANs Algorithm:

1) Find the first minicolumn with a PA neuron. If no such column exists, choose a neuron in the last minicolumn with the minimum number of outgoing connections (MNOCs) and treat it as a PA neuron. Name this first minicolumn with PA neuron FPA minicolumn.
2) Starting from the predecessor minicolumn to FPA
   a) Choose a neuron in this minicolumn that has a link to the PA neuron in FPA and treat it as a PA neuron.
   b) If no such neuron exists, choose a neuron in the predecessor minicolumn with the MNOC neuron and treat it as a PA neuron. This establishes a link between the two PA neurons.

Repeat this step for the new PA neuron, selecting a neuron in its predecessor minicolumn with the MNOCs and treat it as a PA neuron, until no predecessor minicolumn is found.

3) Starting from the PA neuron in the first activated minicolumn, follow the path to the last connected PA neuron in the input sequence and repeat this step until no successor minicolumn is found.
   a) If the successor minicolumn has a PA neuron, link the two PA neurons and follow the path to the last connected PA neuron.
   b) If the successor is a UA minicolumn, choose an MNOC neuron in this minicolumn and treat it as a *PA neuron. Link the two PA neurons and move to* the successor minicolumn.

C. Design Example

Let us illustrate this location of PA neurons by continuing our example. In Fig. 5, the first minicolumn with a PA neuron is M, so according to step 1 of the LPAN algorithm, we name M the FPA minicolumn and move to step 2. In the predecessor minicolumn L, there was no neuron that linked to PA in M. Notice that although there was a link between the fourth neuron in L and the minicolumn M, it did not link to a PA neuron in this minicolumn, so it could not be used.

Following step 2.b of the LPAN algorithm, we choose an MNOC neuron in the minicolumn L and treat it as a PA neuron. The selected MNOC neuron in L is treated as a new PA neuron. This established a new link between the two PA neurons as shown by a dashed line in Fig. 6.

Next, the LPAN algorithm moves back to the minicolumn K. Applying step 2.b of the LPAN algorithm again, we choose a neuron in K with the minimum number of the outgoing connections and link it to the PA neuron in the minicolumn L as shown in Fig. 7.

Since there is no predecessor to K, according to step 3 of the LPAN algorithm, we follow the path from K to the last connected PA neuron in the input sequence which is the neuron Q. Since the successor minicolumn (R) does not have a PA neuron, we follow step 3.b of the LPAN algorithm and choose an MNOC neuron in this minicolumn, and treat it as a PA neuron. This establishes a new link between these two PA neurons in Q and R as illustrated in Fig. 8, and we move to minicolumn R.

Subsequently, following step 3.a of the LPAN algorithm, we link PA neurons in minicolumns R and A and move to the PA neuron in minicolumn I. We finish by applying step 3.b of the LPAN algorithm, which will choose a PA neuron in the minicolumn J and link the PA neurons in I and J as shown in Fig. 8.

This completes the LPAN algorithm and at the same time point II.2 of the LUMAKG algorithm. As a result, we have a sequence of connected PA neurons from the first to the last minicolumn activated by the input sequence. Since there is no successor minicolumn to J, the LUMAKG algorithm moves to III and modifies the synaptic weights between all the established predecessor–successor neurons in the input sequence. Their weights are modified according to the ANAKG algorithm [11].

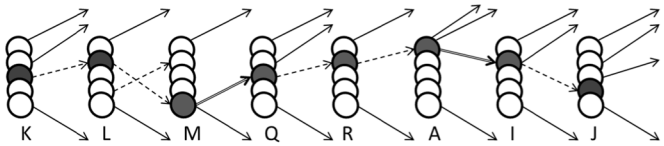

Fig. 8. New connections between the PA neuron in Q and a selected MNOC neuron in the minicolumn R. Additional connections are established between the PA neurons in R and A and between I and a new PA neuron in J.

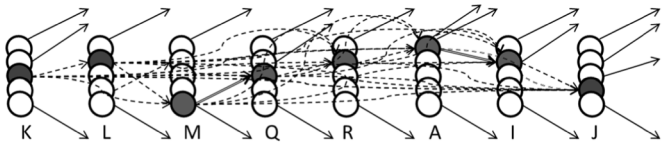

Fig. 9. Modified synaptic connections for the input sequence.

After application of the ANAKG algorithm to modify weights between the selected PA neurons, we will get all the updated links as shown in Fig. 9.

V. Comparative Tests of LUMAKG

We performed several tests to observe the efficiency of learning, memory capacity, and learning resolution for LUMAKG sequential memory, comparing them with similar features of the ANAKG memory and LSTM. A single layer LSTM network with 256 units was created using the TensorFlow library [29] and was used for testing.

A. Test Preparation

The first test is used to compare the resolution of recalled sentences using LUMAKG, ANAKG, and LSTM. To test the recall resolution, the memories were self-organized on an input file containing the text from The Children's Book Test (CBT) [30]. Note that special characters, such as commas and periods, were discarded and not used in training the memories. We read all sentences that were at least ten words long from the database, providing us with over 19 000 sentences with over 9000 unique words. The same sentences were used to obtain LUMAKG, ANAKG, and LSTM memory structures and their respective synaptic connections. After the three memories had been created, their associative memory properties and recall resolution were tested and compared.

To compare how accurately the memories recall stored sequences, we first trained the memories with the first ten words of the sentences. Subsequently, the first six words from each training sequence were used as an input to the LUMAKG, ANAKG, and LSTM memories, and the original test sequences were used as the desired responses. All three memories require only a single presentation of the input data to learn. To observe the effect of increasing the training set size, we started with 100 sentences and gradually increased this size to 10 000 sentences, in increments of 100 sentences for the first 1000 sentences and subsequently in increments of 1000. To illustrate how a number of neurons in minicolumn affects the results, three LUMAKG memory structures with minicolumn sizes 4, 8, and 12 were tested.

B. Network Response Quality Measures

A variety of heuristics and evaluation measures for information retrieval and related tasks have been proposed, for example, answer scoring and/or ranking [31], passage retrieval [32], and evaluating search engines [33]. These evaluation measures require the use of tools such as parsers, and, consequently, are not well suited for evaluation of the responses generated by the LSTM, ANAKG, and LUMAKG memories. Consequently, here we make use of the Levenshtein distance [34], and a new distance measure called RWP based on the evaluation metrics from [35].

1) Levenshtein Distance Quality Measure: The quality of results obtained from the LSTM, ANAKG, and LUMAKG memories was first measured by comparing them with the desired output using the Levenshtein distance [34]. Since we are interested in sequences of words rather than individual characters, the Levenshtein distance measured the number of words that must be deleted, inserted, or substituted in order to transform the source sentence to a target sentence. Each word had a unique symbol in the associative memories, and sequences of such symbols represented the output from each memory.

The Levenshtein distance between two strings a and b (of lengths u and v, respectively) is given by (3)

$$d_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if} \quad \min(i,j)=0 \\ \min \begin{cases} d_{a,b}(i-1,j)+1 & \text{deletion} \\ d_{a,b}(i,j-1)+1 & \text{insertion} \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} & \text{substitution} \end{cases} & \text{otherwise} \end{cases} \tag{3}$$

where $d_{a,b}(i,j)$ is the distance between the first i and j elements of a and b, respectively, and $1_{(a_i \neq b_j)}$ is an characteristic function equal to 0 when $a_i = b_j$ and equal to 1 otherwise. Here, we represent each word in a sentence as a symbol and in computing the Levenshtein distance measure between two sentences, we compare two strings of symbols.

Test I used training and testing sentences as described in Section V-A. We tested the responses of the networks to the same set of inputs, and output sequences obtained by each network were compared with the desired responses using the Levenshtein distance. The larger the Levenshtein distance is, the less similar stored and recalled sequences are, so this distance can be used to compare the quality of the sequential memory.

Fig. 10 shows the mean Levenshtein distances for LSTM, ANAKG, and LUMAKG memories as a function of the number of symbols used in the training data. The test results show that the average of the mean Levenshtein distance between the desired responses and those generated by LSTM and ANAKG memory across all tests was 3.23 and 3.48, respectively, while

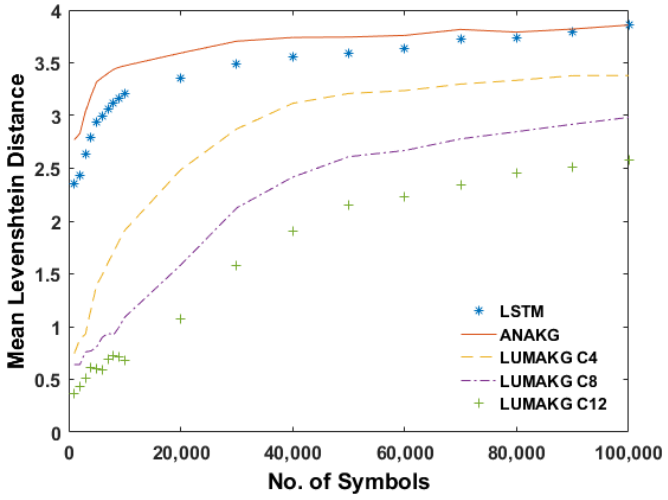

Fig. 10. Plot of mean Levenshtein distances for the LSTM, the ANAKG network, and the LUMAKG networks of different column sizes as a function of the number of symbols.

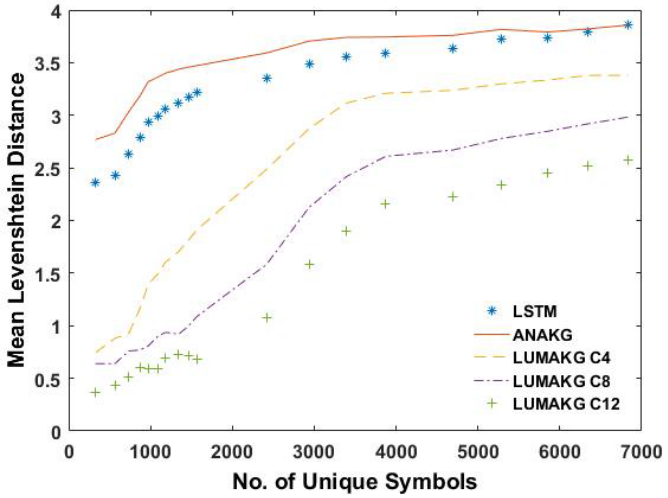

Fig. 11. Plot of mean Levenshtein distances for the LSTM, the ANAKG network, and the LUMAKG networks of different column sizes as a function of the number of unique symbols.

that for the LUMAKG memory was 2.21, 1.65, and 1.30 for column sizes 4, 8, and 12, respectively.

Since the used symbols may be repeated many times in the sentences, the number of unique symbols (words) grows slower than the number of all symbols used in training. Fig. 11 shows the mean Levenshtein distances for the LSTM, ANAKG, and LUMAKG memories as a function of the number of unique symbols used. We see the similar dependence of the distances between stored and restored sequences as on the previous figure. This indicates that as the number of words in the memory grows, it is more difficult to restore the original sequence. Thus, setting some recall standards will determine the memory capacity.

The results obtained from LUMAKG were significantly better than those obtained from both LSTM and ANAKG. The performance of LSTM, while initially better than ANAKG, begins to get worse as network size increases and reaches similar values at 100 000 symbols presented. We can observe that as the network grows in size, the quality of recall expressed by the Levenshtein distance is lower. All types of associative memories showed that they could provide a reasonable output given a limited training data set. However, LUMAKG has the promise to significantly increase both the resolution and storage capacity of the associative knowledge graphs and become a foundation for the semantic memory capable of remembering episodes, making associations and accumulating of knowledge.

2) Reciprocal Word Position: A challenge in evaluating responses of associative spatiotemporal memories, like those based on LSTM and ANAKG, is linked to the difficulty in determining what the correct response is. Thus, the usefulness of Levenshtein distance, a good measure of text similarity, is limited. To address this problem, we designed a new distance measure called the RWP.

The RWP measures the user's effort in extracting the desired response from the output generated by the semantic memory. The RWP distance between two sequences a and b is calculated as follows.

1) Compare the positions of all the words in the desired output sequence (desired response) a with those in the actual memory output sequence (obtained response) b
   a) if the positions of a word in both sequences are the same, the word gets a weight of 1;
   b) if the positions are different by 'n' locations in the tested sequence the word gets a weight of 1/(n+1);
   c) if a word from the desired response does not exist in the obtained response, it gets a weight of 0.
2) The RWP distance equals one minus the sum of the weights of all the words in the desired sequence divided by the maximum of the number of words in the desired and actual sequence

$$d_{\mathrm{RWP}}(a,b) = 1 - \frac{\sum_{i=1}^{k} \frac{1}{|p_{io}-p_{id}|+1}}{(k,l)} \tag{4}$$

where k is the number of words in the desired sequence a, l is the number of words in the obtained sequence b, p_{io} is the position of the word i in the obtained sequence b, and p_{id} is the position of the word i in the desired sequence a.

1) [3)]The RWP distance satisfies the following conditions:

$d_{\mathrm{RWP}}(a,b) = 0 \Leftrightarrow a = b$ identity of indiscernibles
$d_{\mathrm{RWP}}(a,b) = d_{\mathrm{RWP}}(b,a)$ symmetry
$d_{\mathrm{RWP}}(a,c) \leq d_{\mathrm{RWP}}(a,b) + d_{\mathrm{RWP}}(b,c)$ triangle inequality.

The measure is normalized since the lowest value is 0 and the highest is 1, and a lower value of RWP indicates a better match between the sequences. For example, assume that the desired output is "likes cold water" and the generated answer is "cold water likes." Then the second and third words from the desired output are shifted by one position, whereas the first word is shifted by two positions in the generated output, and the resulting RWP distance is $1 - (1/2+1/2+1/3)/3 = 5/9$.

The mean RWP measures for the LSTM, ANAKG, and LUMAKG memories are shown in Fig. 12 as a function of the number of symbols. The test results of applying the

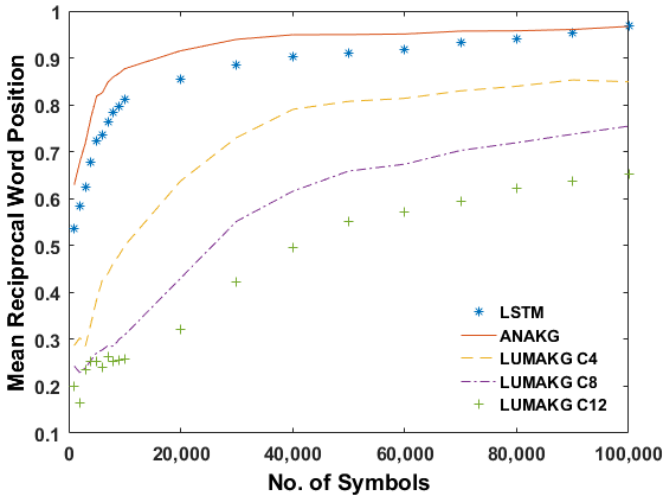

Fig. 12. Plot of mean RWP for the LSTM, the ANAKG network, and the LUMAKG networks of different column sizes as a function of the number of symbols.

RWP distance to the different memory outputs show that the average RWP distance between the desired response and the one generated by LSTM, and ANAKG memory was 0.81 and 0.87, respectively, while the average distance for the LUMAKG memory was 0.58, 0.45, and 0.38 for column sizes 4, 8, and 12, respectively.

These results also show that the performance of LUMAKG-based semantic memory is better than LSTM, which is better than ANAKG-based semantic memory, and its relative recall quality over both LSTM and ANAKG increases as the column size increases.

C. Recall Quality and Memory Capacity

The third type of tests was to show the dependence of the *memory capacity* on the number of neurons in each minicolumn and the number of objects (individual words) stored. The memory capacity can be established for a specific level of the *recall quality* (R_{QL}), where the recall quality for the results obtained with the Levenshtein distance is defined as

$$R_{\mathrm{QL}} = \underset{S}{\mathrm{mean}}\left(1 - \frac{D_{\mathrm{Ls}}}{\max(L_{\mathrm{s1}}, L_{\mathrm{s2}})}\right) \tag{5}$$

where D_{Ls} is the average Levenshtein distance divided by the maximum length (number of words) of the stored and recalled sentences; the average is taken over all the test sentences. The R_{QL} value is between 0 and 1, with 1 representing a perfect recall and 0, a completely wrong recall.

We studied R_{QL} as a function of the number of objects used in sentences—objects are individual words and each repetition counts as a new object. In general, the larger the size of the associative memory is, the lower its recall quality is. In our test, we set the recall quality threshold $T_{\mathrm{RQ}} = 70\%$ and tested at which the number of objects stored $R_{\mathrm{QL}} < T_{\mathrm{RQ}}$ to determine the *memory capacity*.

Similarly, we may establish the memory capacity using the recall quality based on RWP distance measure. Since distance based on RWP is already normalized, we can define recall

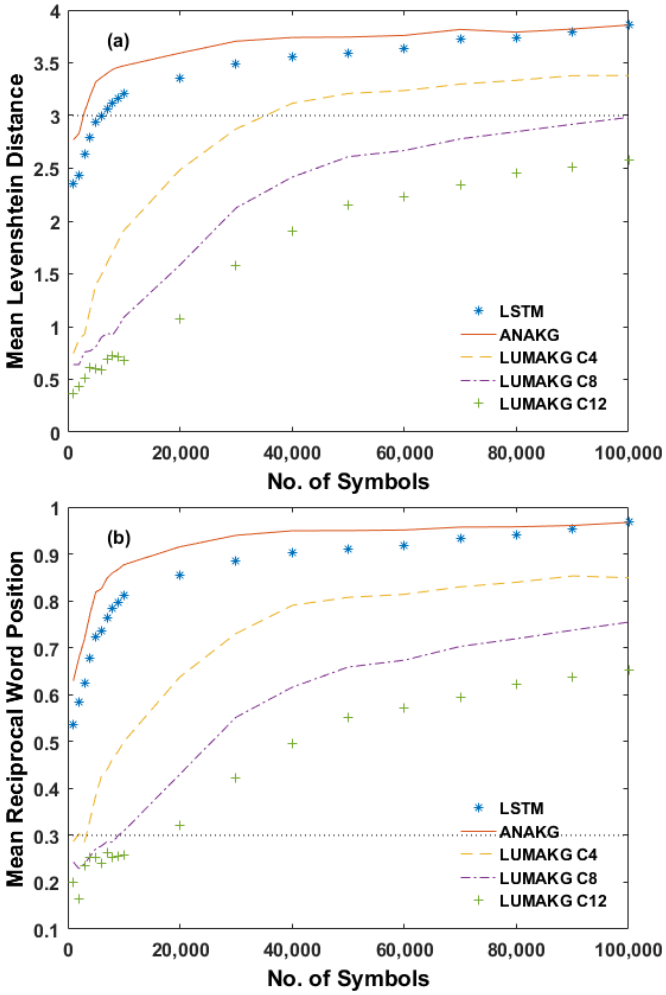

Fig. 13. Plot of the mean (a) Levenshtein distances and (b) RWP, for the LSTM, the ANAKG networks, and the LUMAKG networks of different column sizes as a function of the number of symbols. The reference line at recall quality threshold of 70% was added.

quality as

$$R_{\mathrm{QL}} = \underset{S}{\mathrm{mean}}\,(1 - d_{\mathrm{RWPs}}). \tag{6}$$

There is a strict correspondence between the threshold value set to establish the memory capacity and the distance measure used. In addition, although the two distance measures look similar, recall quality based on these measures are not.

For instance, from (5), it follows that a recall quality threshold of 70% equals a mean Levenshtein distance of 3, but this roughly corresponds to RWP distance equal 0.75 with the recall quality threshold 0.25. Considering that RWP is scaled between 0 and 1 it may be more convenient to specify memory capacity based on this measure.

Fig. 13(a) and (b) repeats results from Figs. 10 and 12, respectively. A reference line in Fig. 13(a) is used to determine memory capacity for a Levenshtein distance equivalent to the recall quality threshold of 70%. The test results based on LSTM, ANAKG, and LUMAKG with minicolumn sizes of 4 show that the Levenshtein distance is less than 3 for networks with up to 6075, 2820, and 35 230 symbols, respectively.

Similarly, the reference line in Fig. 13(b) shows that LUMAKG networks with minicolumn sizes of 4, 8, and

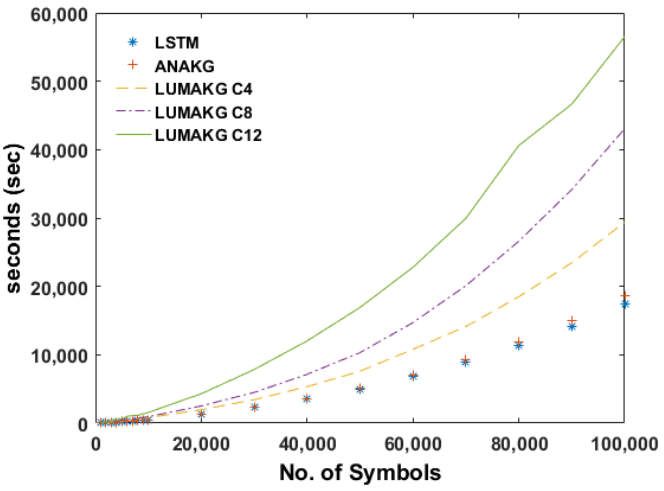

Fig. 14. Learning times of the LSTM, ANAKG network, and the LUMAKG network of different column sizes as a function of the number of symbols.

12 have RWP less than 0.3 for networks with about 3290, 8995, and 16 000 symbols, respectively. Memory capacity based on this RWP distance would be zero for LSTM and ANAKG. To make them similar as obtained in the case of the Levenshtein distance, we should change the threshold of average RWP position to about 0.7, which correspond to recall quality value of 30%.

In summary, the results show that the recall quality and memory capacity of LUMAKG networks increases as the size of minicolumn increases and this is better than ANAKG network. Recall the quality and memory capacity of LSTM is slightly better than ANAKG but not as good as LUMAKG.

D. Computational Complexity

The fourth type of tests was performed to determine the computational complexity of LUMAKG memory in comparison to LSTM and ANAKG memories. We tested the time needed to create the associative memory as a function of the number of objects. The results presented in Fig. 14 show the learning time for LSTM, ANAKG, and LUMAKG as a function of all objects in the data set. We see that computational cost for LUMAKG is between 60% and 200% higher than for ANAKG (depending on the minicolumn size), whereas that for ANAKG is about 6% higher than LSTM.

The overall increase in the simulation time shows less than the quadratic relationship to the size of the training data set, which allows storing a large number of sequences in LUMAKG memory. In addition, an increase in the simulation time due to an increase in the number of neurons in each minicolumn is less than linear. Again, this is a desired property of the developed method, since minicolumns in the human brain contain upward of 100 neurons each. Tests were performed on a general purpose computer (i7-4790, 3.6 GHz, 16-GB RAM).

The number of neurons in LUMAKG memory is k times larger than in ANAKG memory, where k is the number of neurons in each minicolumn (in our tests $k = 4$, 8, or 12). However, the number of synapses does not grow as fast since the number of associative links between all neurons corresponds to the number of transitions between various words. These transitions are just spread over the larger number of neurons. Although the training time is greater for LUMAKG than for ANAKG due to the larger number of neurons and the need of finding predecessor and successor for each element of the training sequence, the quality of results points to better properties of LUMAKGs, which make them more suitable to develop short-term associative memories.

One way of addressing the issue of training time would be to consider the requirements of the application (e.g., performance vs. computational time requirements) and appropriately determine the minicolumn size. To do this effectively, the above tests need to be repeated, with different data sets and with the randomized order of sequences. There are also some implementation level options for addressing the issue of training time. These include parallelization of the algorithms, and the neuron model to efficiently use multiple cores or CPUs, offload operations to GPUs, and optimization of libraries used.

VI. Conclusion

LUMAKG memory supports continuous online learning, self-organization without supervised learning, context-based predictions, and is capable of recognizing time-domain sequences correctly. In our work, we build on two powerful mathematical models of associative memories HTM and ANAKG. Both methods were developed and tested successfully on sequential data, self-organizing the memory structure and extracting associative information. Our work combined the two models taking advantage of what they offer best—the large capacity of the memory as demonstrated in [12], [26], and [36] and auto-associative properties with fast implementation in pulsing neurons as demonstrated in [11] and [14]. In our work, each column represents a separate symbol or feature, and thus the number of columns in the network equals the number of distinct symbols. In ANAKG network, the number of synaptic connections equal to a number of pairs of presynaptic and postsynaptic neurons in the training sequences. In LUMAKG, this number is only slightly larger, since columns are connected considering the broader context of each succession.

Test of recall quality, memory capacity, and computational complexity of LUMAKG networks were performed and compared to similar tests of ANAKG and LSTM memories. The effect of varying the number of neurons in minicolumns on memory performance was also studied. LUMAKG shows better ability to recall sequences stored in the memories than LSTM or ANAKG. Using the Levenshtein distance and another quality measure, we also show that LUMAKG memory has higher capacity and better resolution for short-term memory recall. Future work will extend LUMAKG to a distributed representation of all symbols stored in the memory which will significantly increase its storage capacity. Further studies will also be performed on different types of input data, for example, image and audio, to obtain an assessment of the network properties in different applications.

References

[1] J. R. Binder and R. H. Desai, "The neurobiology of semantic memory," *Trends Cogn. Sci.*, vol. 15, no. 11, pp. 527–536, Nov. 2011.

[2] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Discrete search leading continuous exploration for kinodynamic motion planning," in *Proc. Robot., Sci. Syst. (RSS)*, Atlanta, GA, USA, Jun. 2007, pp. 326–333.

[3] L. R. Squire, "Memory systems of the brain: A brief history and current perspective," *Neurobiol. Learn. Memory*, vol. 82, no. 3, pp. 171–177, Nov. 2004.

[4] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1998.

[5] M. A. Kramer, "Autoassociative neural networks," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 313–328, 1992.

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

[7] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 5528–5531.

[8] J. Weston, S. Chopra, and A. Bordes, "Memory networks," Oct. 2014, *arXiv:1410.3916*. [Online]. Available: https://arxiv.org/abs/1410.3916

[9] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé, "A neural network for factoid question answering over paragraphs," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Oct. 2014, pp. 633–644.

[10] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," Oct. 2014, *arXiv:1410.5401*. [Online]. Available: https://arxiv.org/abs/1410.5401

[11] A. Horzyk, "How does generalization and creativity come into being in neural associative systems and how does it form human-like knowledge?" *Neurocomputing*, vol. 144, pp. 238–257, Nov. 2014.

[12] J. Hawkins, S. Ahmad, and D. Dubinsky, "Cortical learning algorithm and hierarchical temporal memory," Numenta, Redwood City, CA, USA, Tech. Rep. Version 0.2.1, Sep. 2011, pp. 1–68. [Online]. Available: http://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf

[13] A. Horzyk, J. A. Starzyk, and J. Graham, "Integration of semantic and episodic memories," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 12, pp. 3084–3095, Dec. 2017.

[14] A. Horzyk, "Deep associative semantic neural graphs for knowledge representation and fast data exploration," in *Proc. KEOD*, 2017, pp. 67–79.

[15] A. Horzyk, J. A. Starzyk, and Basawaraj, "Emergent creativity in declarative memories," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, Dec. 2016, pp. 1–8.

[16] A. Horzyk, "How does human-like knowledge come into being in artificial associative systems," in *Proc. 8th Int. Conf. Knowl., Inf. Creativity Support Syst.*, Krakow, Poland, 2013, pp. 1–12.

[17] A. Horzyk and J.A. Starzyk, "Associative fine-tuning of biologically inspired active neuro-associative knowledge graphs," in *Proc. IEEE Symp. Series Comput. Intell. (SSCI)*, pp. 2068–2075, Nov. 2018.

[18] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, 2016.

[19] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 609–616.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[21] J. M. McFarland, Y. Cui, and D. A. Butts, "Inferring nonlinear neuronal computation based on physiologically plausible inputs," *PLoS ONE*, vol. 9, no. 7, Jul. 2013, Art. no. e1003143.

[22] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[23] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.

[24] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[25] E. M. Izhikevich, "Polychronization: Computation with spikes," *Neural Comput.*, vol. 18, no. 2, pp. 245–282, 2006.

[26] J. Hawkins and S. Ahmad, "Why neurons have thousands of synapses, a theory of sequence memory in neocortex," *Frontiers Neural Circuits*, vol. 10, p. 23, Mar. 2016.

[27] *GitHub—Numenta/Nupic: Numenta Platform for Intelligent Computing is an Implementation of Hierarchical Temporal Memory (HTM), a Theory of Intelligence Based Strictly on the Neuroscience of the Neocortex*. Accessed: Jul. 12, 2018. [Online]. Available: https://github.com/numenta/nupic

[28] O. Sporns, G. Tononi, and R. Kötter, "The human connectome: A structural description of the human brain," *PLoS ONE*, vol. 1, no. 4, p. e42, Sep. 2005.

[29] *TensorFlow*. Accessed: May 27, 2018. [Online]. Available: https://www.tensorflow.org/

[30] *bAbI—Facebook Research*. Accessed: Jun. 2, 2018. [Online]. Available: https://research.fb.com/downloads/babi/

[31] S. K. Ray, S. Singh, and B. P. Joshi, "A semantic approach for question classification using WordNet and Wikipedia," *Pattern Recognit. Lett.*, vol. 31, no. 13, pp. 1935–1943, Oct. 2010.

[32] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, "Quantitative evaluation of passage retrieval algorithms for question answering," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2003, pp. 41–47.

[33] S. Büttcher, C. L. A. Clarke, and G. V Cormack, *Information Retrieval: Implementing and Evaluating Search Engines*. Cambridge, MA, USA: MIT Press, 2016.

[34] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Phys. Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966.

[35] D. R. Radev, H. Qi, H. Wu, and W. Fan, "Evaluating Web-based question answering systems," in *Proc. 3rd Int. Conf. Lang. Resour. Eval. (LREC)*, May 2002, pp. 1–4.

[36] J. Hawkins, D. George, and J. Niemasik, "Sequence memory for prediction, inference and behaviour," *Phil. Trans. Roy. Soc. B, Biol. Sci.*, vol. 364, no. 1521, pp. 1203–1209, May 2009.

Basawaraj received the B.E. degree in electronics and communication engineering from the S.L.N. College of Engineering, Raichur, India, in 2000, and the M.Tech. degree in digital electronics and communication from the Ramaiah Institute of Technology, Bengaluru, India, in 2004. He is currently pursuing the Ph.D. degree with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH, USA.

His current research interests include embodied machine learning, self-organizing associative memories, and mathematical modeling of concept learning, and categorization behavior.

Janusz A. Starzyk (LSM'16) received the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the Warsaw University of Technology, Warsaw, Poland, in 1971 and 1976, respectively, and the Habilitation degree in electrical engineering from the Silesian University of Technology, Gliwice, Poland.

He has been an Assistant Professor with the Institute of Electronics Fundamentals, Warsaw University of Technology. He has been a Professor of electrical engineering and computer science with Ohio University, Athens, OH, USA. Since 2007, he has also been the Head of the information systems applications, University of Information Technology and Management, Rzeszów, Poland. His current research interests include embodied machine intelligence, motivated goal-driven learning, self-organizing associative spatiotemporal memories, active learning of sensory–motor interactions, machine consciousness, and applications of machine learning to autonomous robots and avatars.

Adrian Horzyk (SM'18) received the M.S. degree in computer science from Jagiellonian University, Kraków, Poland, in 1997, and the Ph.D. and Habilitation degrees in computer science from the AGH University of Science and Technology, Kraków, in 2001.

He is currently an Associate Professor with the Faculty of electrical engineering, automatics, computer science, and biomedical engineering, Department of Biocybernetics and Biomedical Engineering, AGH University of Science and Technology. His current research interests include the development of knowledge-based models and methods of artificial intelligence and computational intelligence, associative and spiking models of neurons and their networks, self-developing semantic associative memories, new machine learning strategies and techniques, data science, data mining, knowledge engineering methods, and cognitive systems.

Dr. Horzyk has been a Co-Founder and a member of the Polish Association of Artificial Intelligence since 2009 and a Board Member of the Polish Neural Network Society (PTSN) since 2011. He has been the Deputy Team Leader of the CERN Alice experiments and projects with the AGH University of Science and Technology since 2017.