Lumped Mini-Column Associative Knowledge Graphs

Basawaraj[1], Janusz A. Starzyk[1,2]
[1] *School of EECS, Ohio University, Athens, OH 45701, USA*
[2] *University of Information Technology and Management in Rzeszow*
Rzeszow, Poland
{starzykj,basawaraj.basawaraj.1}@ohio.edu

Adrian Horzyk
Dept. Automatics and Biomedical Engineering
AGH University of Science and Technology in Krakow
Mickiewicza Av. 30, 30-059 Krakow, Poland
horzyk@agh.edu.pl

***Abstract* — This paper presents an extension of active neural associative knowledge graphs (ANAKG) to their mini-column form where each symbol is represented several times. We demonstrate that this new associative memory organization preserves all properties of ANAKG memories like storage of knowledge based on spatio-temporal input sequences, while increasing recall quality, memory capacity, and its resolution for short-term memory recall. The implemented model combines ANAKG associative neuron idea with the idea of a hierarchical temporal memory that uses a mini-column form of symbol representation. Performed tests confirm our claims of higher resolution and higher memory capacity of the new associative knowledge graph.**

Keywords — associative semantic memory, associative neurons, associative connections, mini-column structure, knowledge representation.

I. Introduction

Semantic memory (SM) is a repository of knowledge in a cognitive system, and its structure gradually emerges from the learning process [1]. Neurons in the semantic memory build synaptic connections between associated concepts representing their relationships between each other and cognitive agent objectives [2].

Classical neural network memory models such as associative memory networks [3] aim to provide content-addressable memory, capable of retrieving stored data from only a small input sample. Other solutions to retrieving the memory information include gradient based recurrent neural networks (RNN) [4]. These models are trained to predict the next input word after reading the first few words. In [5] authors proposed an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts. Similar to RNNs this algorithm is trained to predict words in a document given an input context. In [6] authors describe a new class of learning models called memory networks. Memory networks combine input content with dynamic knowledge base stored in 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 desired response.

Recently researchers have focused on contextual question answering (QA) to satisfy popular demand on services based on voice recognition and large knowledge bases like Wikipedia. Direct approaches to QA like string matching are very ineffective, and solutions that include recursive neural networks like QANTA [7] are becoming popular. Neural Turing machines (NTM) combine the concept of neural network learning and classical Turing machines to retrieve context based input information [8]. NTMs can learn simple algorithms from training data and generalize them to non-training data. Comparing to RNN long short-term memory [4] NTMs show better accuracy over longer sequences in recall and copy tasks.

Semantic memory aggregates representation of the training data and forms a context searchable knowledge base. It is obtained by binding the semantic contexts for all trained objects and linking their neuronal representations together. The SM can be built using active neural associative knowledge graph (ANAKG) based on associative neuron models presented in [9]. The knowledge graphs are obtained dynamically 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 previously activated neurons create inhibitive connections to the incorrectly activated neurons. The gradual time-spread relaxation of ANAKG neurons enables them to represent sequences of objects in their previous contexts.

An ANAKG based SM can associate distant time events in order to put them in a wider context. Moreover, the SM can trigger 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

SM can generalize knowledge gained during the adaptation process based on presented training data. The ANAKG based SM can generate new responses according to the new contexts of recollection that were not previously taught [10]. The generalization is possible thanks to the association and aggregations of data, symbols, features, objects, and subsequences which typically occur in training data.

In this paper, we present a generalization of ANAKGs to their mini-column form known as a lumped mini-column associated knowledge graph (LUMAKG). In LUMAKG, each symbol is represented several times following the idea of a mini-column organization presented in [11]. LUMAKG uses the same pulsing neuron model as ANAKG and similar self-organization principles. The major difference is in its columnar organization and selection of synaptic connections between neurons.

II. LUMAKG Organization and Principles

The columnar organization of associative memory was first proposed by J. Hawkins et al. [11] where they introduced cortical learning algorithms in which mini-columns were used to store sequential information in hierarchical temporal memory (HTM). Since then HTMs were further developed, and their properties were analyzed and tested. In [12] using mini-column model, the authors show that it is 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 like echo state networks [13] or long short term memory [14], however, they also show some drawbacks like larger sensitivity to temporal noise than the long short term memory [12]. ANAKG memories do not have this drawback of HTM networks as they can tolerate temporal noise. Thus improving ANAKG by introducing a mini-column structure to its architecture provides a better associative memory capable of storing spatio-temporal relations between data.

ANAKG uses a pulsing neuron model that uses spatio-temporal relationship between data, similar to the popular spiking neuron models [15]. Spiking neurons are biologically motivated and produce patterns similar to biological neurons. Several computationally efficient models of spiking neurons have been developed [16]. Networks of spiking neurons spontaneously self-organize into groups and generate patterns of polychronous activity, and this property is believed to be necessary for cognitive neural computations, symbol grounding, attention, and consciousness [17]. ANAKG achieves similar properties to spiking neurons by using a much simpler neuron model and self-organization principles [18]. LUMAKG maintains these properties of ANAKG while increasing its recall quality, memory capacity, and resolution.

Following HTM organization, we replace each neuron in the ANAKG with a small mini-column. Each mini-column has five neurons, and all mini-column neurons represent one unique symbol (e.g. a single word). While the neurons in a mini-column all represent the same symbol, the inputs, outputs, and their synaptic connections (weights) are different for each neuron. Using the principles from HTM [11], the inputs and outputs are distributed across the neurons in the mini-column so that multiple sequences can be represented using the same set of mini-columns. Individual neurons in each mini-column use the ANAKG algorithm to obtain associative connections and their weights.

Like in HTM, LUMAKG mini-columns 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 cells/neurons can fire even without sensory input stimulation. In the predictive mode, neuron's activation from the lateral input is used to complete the sequence. During learning of new sequences, prediction and input activation should match for learning (changing synaptic weights) to take place.

The LUMAKG network structure is obtained dynamically. New mini-columns 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 mini-column is added, and at least one of its neurons is linked to another mini-column. Organizing principles of LUMAKG are as follows:

A. Duplicate each symbol five times to form individual symbol **mini-columns**.
B. If a node in a mini-column is activated above the threshold from associative connections, it is called to be in a **predictive mode**.
C. An input activates either all nodes in a given mini-column that are in the predictive mode or the whole mini-column if no node is in a predictive mode.
D. Activated nodes that were in a predictive mode are in **predicted activation (PA)**.
E. An activated mini-column without any node in a predictive mode has all nodes in **unpredicted activation (UA)**.
F. **Synaptic connections weights** are changed between activated nodes in predecessor and successor mini-columns.

III. LUMAKG Algorithm and its Illustration

The LUMAKG algorithm was defined on the basis of the general organizing principles described in the previous section:

I. Read the consecutive elements of the input sequence to activate corresponding mini-columns:

1. Check if the symbol from the input sequence is represented by a mini-column.
2. If not add a new mini-column; all nodes of this new mini-column are in unpredicted activation.

II. Establish predecessor-successor nodes in all activated mini-columns in the input sequence:

3. For each consecutive activated mini-column, activate nodes in the mini-column that corresponds to the input symbol, according to point C of the organizing principles. Typically, the first activated mini-column has no PA nodes, unless it is considered in a broader context of associative learning. Thus typically, all nodes in the first activated mini-column are in unpredicted activation.
4. Find the first mini-column with a PA node. If no such column exists choose a node in the last mini-column with

a minimum number of outgoing connections and treat it as a PA node. Name this first mini-column with PA node FPA (first predicted activation) mini-column.

5. Starting from the predecessor mini-column to FPA:
   a. Choose a node in this mini-column that has a link to the PA node in FPA and treat it as a PA node.
   b. If no such node exists, choose a node in the predecessor mini-column with the minimum number of outgoing connections and treat it as a PA node. This establishes a link between the two PA nodes.

   Repeat this step for the new PA node, selecting a node in its predecessor mini-column with the minimum number of outgoing connections and treating it as a PA node, until no predecessor mini-column is found.
6. Starting from the successor mini-column to FPA repeat until no more successor mini-column is found:
   a. If the successor mini-column has a PA node, link the two PA nodes and move to the successor mini-column.
   b. If the successor is a UA mini-column, choose a node in this mini-column with the minimum number of outgoing connections and treat it as a PA node. Link the two PA nodes and move to the successor mini-column.

III. Update synaptic weights in the synaptic connections between all predecessor successor nodes:

7. The algorithm updates all the synaptic weights between all PA nodes in predecessor and successor mini-columns according to rules developed for ANAKG [9].

Since the LUMAKG algorithm has a convoluted process of modifications of synaptic connections, we use an example to illustrate this algorithm. For simplicity, we assume that a PA node makes a single prediction and that at most a single PA node is activated in any mini-column. While the algorithm is not limited to such cases, these assumptions will simplify an example illustration of how the LUMAKG algorithm works. For simplicity of graphical illustration, a mini-column is represented by a single node with double lines as shown in Fig. 1.

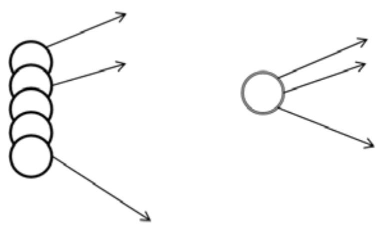

Fig. 1. A mini-column and its simplified symbol

The corresponding synaptic connections start at various nodes in the mini-column as we can observe in Fig. 1. In particular, the predecessor nodes are hidden in the simplified symbol, so we cannot see which of the mini-column neurons is the predecessor node. This can be observed only in the full view of the resulting structure as shown in Fig. 7.

Let us assume that the sequence A, B, C, D, E was inputted to the LUMAKG memory and activated the corresponding mini-columns as shown in Fig. 2.

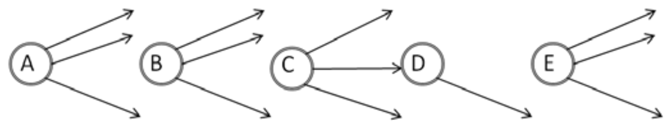

Fig. 2. Activated mini-columns.

Out of these activated mini-columns, only mini-column D had a PA node. According to step 5 of the LUMAKG algorithm, we name D the FPA mini-column and move to step 6. In the predecessor mini-column C, we choose the node that linked to the PA node in D and name it as a new PA node. Next, we move to the predecessor mini-column C and according to step 6.b. choose a node in mini-column B with a minimum number of outgoing connections and treat it as a PA node.

This establishes a new link between B and C as shown by a dashed line in Fig. 3.

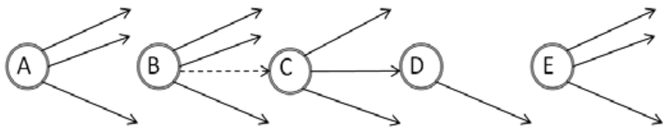

Fig. 3. A new connection between UA mini-column B and a PA node in mini-column C.

The selected node in B with the minimum number of outgoing connections is treated as a new PA node, and the algorithm moves back to mini-column A. Applying step 6.b. again we choose a node in A with the minimum number of outgoing connections and link it to the PA node in B as shown in Fig. 4.

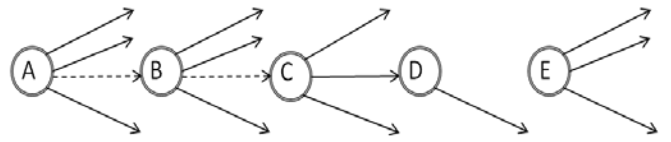

Fig. 4. A new connection between UA mini-column A and a PA node in mini-column B.

Since there is no predecessor to A, we move back to mini-column D. Since there is no PA node in mini-column E, then according to 7.b. we choose the node in mini-column E with the minimum number of outgoing connections and treat it as a PA node. This establishes the link between the two PA nodes in mini-columns D and E as illustrated in Fig. 5 and we move to mini-column E.

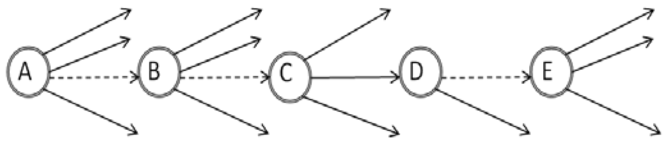

Fig. 5. A new connection between PA node in UA mini-column D and a selected node with the minimum number of outgoing connections in mini-column E.

Since there is no successor mini-column to E, the LUMAKG algorithm moves to 8 and modifies the synaptic weights between all PA nodes in established predecessor and successor mini-columns according to the ANAKG algorithm [9].

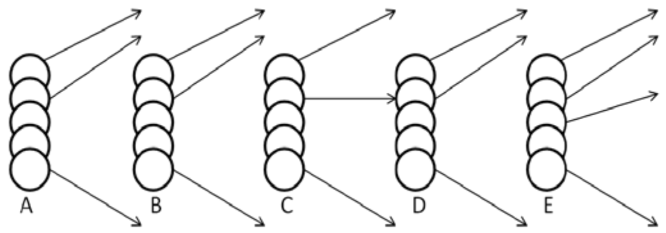

Fig. 6.a. Activated mini-columns.

Fig. 6 shows an example view of the mini-column structure that corresponds to the processed input sequence at the beginning (Fig. 6.a.) and at the end (Fig. 6.b.) of step II of LUMAKG algorithm. In each mini-column, gray color is used to represent PA neurons that will have their synaptic connection weights changed according to [9].

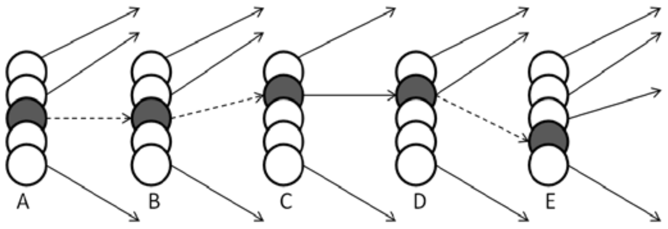

Fig. 6.b. Activated mini-columns with added links.

After application of the ANAKG algorithm to modify weights between the selected PA nodes, we will get all the updated links as shown in Fig. 7.

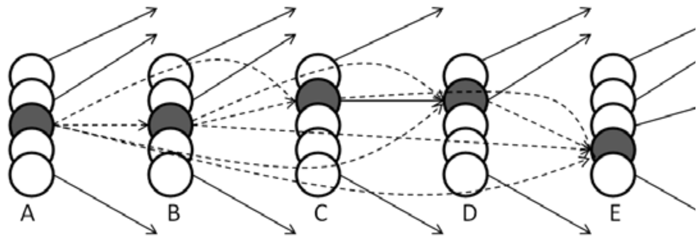

Fig. 7. Modified synaptic connections for the input sequence.

IV. Comparative Tests of LUMAKG

Tests were performed to observe the efficiency of learning, memory capacity, and learning resolution for LUMAKG sequential memory in comparison with ANAKG memory. Tests of both methods were performed on the same equipment and using the same software environment. In fact, LUMAKG was based on a modified ANAKG tool.

A. *Resolution of Recalled Sentences*

The first test is used to compare the resolution of recalled sentences using ANAKG and LUMAKG. To test the recall resolution, the memories ANAKG and LUMAKG, were self-organized on an input file containing the following sentences from *The Golden Bird* tale from Grimm's Fairy Tales [19]:

1. The king had a beautiful garden, and in the garden stood a tree.
2. The tree bore golden apples, apples that were always counted.
3. About the time when the apples grew ripe, it was found that every night one apple was gone.
4. The king was angry at an apple going missing every night.
5. The king ordered his gardener to keep watch all night under the tree.
6. The first day the gardener asked his eldest son to keep watch.
7. About midnight he fell asleep, and in the morning another of the apples was missing.
8. The second day the gardener asked his second son to keep watch.
9. At midnight he too fell asleep, and in the morning another of the apples was missing.

Note that special characters, e.g. commas, periods, etc., were discarded and not used in training the memories. After the memories had been created, their associative properties and recall resolution were tested and compared with results obtained from ANAKG. Several sequences related to nine training sequences were used as inputs to both memories, and their recalls were observed. The results are shown in Table 1.

Table 1. Input and output sequences generated by ANAKG and LUMAKG.

Input sequence	**ANAKG memory output**	**LUMAKG memory output**	**Desired output**
What did the king had?	The king had a beautiful garden	The king had a beautiful garden	The king had a beautiful garden
What stood in the garden?	**Stood a in the tree the garden**	**Stood in the garden stood a tree**	**In the garden stood a tree**
Why was the king angry?	**Was the king angry at an apple missing**	**Was the king was angry at an apple missing**	**The king was angry at an apple going missing**
What were always counted?	Were always counted	Were always counted	Apple that were always counted
What did the king order his gardener?	**the king his gardener**	**The king was his gardener to keep watch**	**The king ordered his gardener to keep watch**
What was missing in the morning?	**Was missing in the morning another of the apple was**	**Was missing in the morning another of the apple was missing**	**In the morning another of the apple was missing**

The results show that LUMAKG provides more meaningful answers than ANAKG (boldface rows in Table 1) considering what was stored in the network as a result of the training sequence. The inputs sentences used in this test were not specifically tailored to benefit one method over another. As we can see, in no case ANAKG provided answers more meaningful than LUMAKG.

B. *Testing a Large Data Set*

The next test involved comparing the performance of LUMAKG and ANAKG memories when the number of sentences in the training set is increased. The two memories were trained with the complete story (The Golden Bird) that had over 2500 words, with over 500 unique words in 78 training sentences. The same input sequences were then applied to both networks. The results are shown in Table 2.

The results show, similar to Test 1, that the answers provided by LUMAKG are more meaningful than those provided by ANAKG. Note that while there is some decrease in quality of LUMAKG answers the decrease is more profound in the case of ANAKG memory. This shows the robustness of LUMAKG memory that is a result of the applied mini-column structure and related associative learning.

TABLE 2. INPUT AND OUTPUT SEQUENCES FOR LARGER NUMBER OF TRAINING INPUT SEQUENCES.

Input sequence	**ANAKG memory output**	**LUMAKG memory output**	**Desired output**
What did the king had?	**What did the king had**	**What did the king had a beautiful garden**	**The king had a beautiful garden**
What stood in the garden?	**What stood in the garden**	**What stood in the garden stood a tree**	**In the garden stood a tree**
Why was the king angry?	**Why should was the king angry at**	**Why should was the king angry at an apple**	**The king was angry at an apple going missing**
What were always counted?	What were always counted	What were always counted	Apple that were always counted
What did the king order his gardener?	**What did the king his gardener**	**What did the king his gardener to keep watch**	**The king ordered his gardener to keep watch**
What was missing in the morning?	**What was missing in the morning**	**What was missing in the morning another of the apple**	**In the morning another of the apple was missing**

C. *Network Response Quality Measures*

A variety of heuristics and evaluation measures for various information retrieval and related tasks have been proposed and studied, e.g. answer scoring and/or ranking [20], passage retrieval algorithms [21], and evaluating search engines [22]. 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 ANAKG and LUMAKG memories. Consequently, here we make use of Levenshtein distance [23], and a new distance measure called reciprocal word position based on the evaluation metrics from [24].

Levenshtein Distance Quality Measure

The quality of results obtained from ANAKG and LUMAKG memories were first measured by comparing them to desired output using Levenshtein distance [23]. 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 (1):

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

where $d_{a,b}(i,j)$ is the distance between the first *i* and *j* elements of *a* and *b* respectively. Here we represent each word in a sentence as a symbol and in computing the Levenshtein distance measure and comparing two sentences, we compare two stings of symbols. The Levenshtein distance was applied to the results of Test I that used only 9 training sentences and Test II that used 78 training sentences. In both cases we tested the networks responses to the same set of inputs and network output sequences were compared with the desired responses. The results are shown in Table 3.

TABLE 3. LEVENSHTEIN DISTANCE TO THE DESIRED OUTPUT IN BOTH TYPES OF MEMORIES.

Input sequence	**ANAKG**		**LUMAKG**	
	Test I	Test II	Test I	Test II
What did the king had?	0	5	0	2
What stood in the garden?	6	5	1	2
Why was the king angry?	3	8	2	6
What were always counted?	2	2	2	2
What did the king order his gardener?	4	6	1	3
What was missing in the morning?	3	8	2	5

As we can see, all the results obtained from LUMAKG were either better or the same as those obtained from ANAKG. Both types of associative memories showed that they could provide a reasonable output given a limited training data set. However, LUMAKG has a 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. Both memories require only a single presentation of the input data to learn.

The desired outputs in the above test share some words/symbols with the input sequence. This is a consequence of forming grammatically valid sentences or sequence of words. However, multiple grammatically valid sentences that represent the same answer can possibly be generated, thus potentially limiting the usefulness of the Levenshtein distances measured above. The above distance measures can be improved by disregarding the words/symbols from the input sequence that are present in the desired output and the outputs from the ANAKG and LUMAKG memories. The main reason for such omission is that neurons associated with these words are directly activated by the input sequence, so their removal simplifies assessment of the network generated response. The resulting outputs are shown in Tables 4 and 5. The results of applying the Levenshtein distance measure to the resulting output sequences so obtained are shown in Table 6 and show that performance of LUMAKG was the same or better than that of ANAKG. These results also show that while both ANAKG and LUMAKG can provide reasonable responses when trained with a small dataset the performance of LUMAKG is considerably better when there are a large number of sequences in the training set.

Reciprocal Word Position

A difficulty in evaluating responses or answers of semantic memories, like those based on ANAKG and LUMAKG, is the difficulty in determining what is the correct response from associative spatio-temporal memory? Thus, Levenshtein distance while a good measure of text similarity is, in this case, limited. Hence, we designed a new quality measure called reciprocal word position (RWP).

RWP measures user's effort in extracting the desired response from the output generated by the semantic memory. RWP is calculated as follows:

Compare the positions of all the words in the desired output to those in the actual memory output,

a) if the positions are the same the word gets a weight of 1;
b) if the positions are different by 'n' words the word gets a weight of 1/(n+1);
c) if a word does not exist it gets a weight of 0;
d) RWP equals to 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 outputs.

The measure is normalized since the lowest value is 0 and the highest is 1 and a higer value of RWP indicate better match between the sequences. For example, assume 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 is (1/2+1/2+1/3)/3 = 4/9.

TABLE 4. INPUT SEQUENCES AND OUTPUT SEQUENCES, DISREGARDING INPUT SYMBOLS, GENERATED BY ANAKG AND LUMAKG.

Input sequence	ANAKG memory output	LUMAKG memory output	Desired output
What did the king had?	a beautiful garden	a beautiful garden	a beautiful garden
What stood in the garden?	a tree	a tree	a tree
Why was the king angry?	at an apple missing	at an apple missing	at an apple going missing
What were always counted?			Apple that
What did the king order his gardener?		was to keep watch	to keep watch
What was missing in the morning?	another of apple	another of apple	another of apple

TABLE 5. INPUT SEQUENCES AND OUTPUT SEQUENCES, DISREGARDING INPUT SYMBOLS, FOR LARGER NUMBER OF TRAINING INPUT SEQUENCES.

Input sequence	ANAKG memory output	LUMAKG memory output	Desired output
What did the king had?		a beautiful garden	a beautiful garden
What stood in the garden?		stood a tree	a tree
Why was the king angry?	should at	should at an apple	at an apple going missing
What were always counted?			Apple that
What did the king order his gardener?		to keep watch	to keep watch
What was missing in the morning?		another of apple	another of apple

TABLE 6. LEVENSHTEIN DISTANCE TO THE DESIRED OUTPUT, DISREGARDING INPUT SYMBOLS, IN BOTH TYPES OF MEMORIES.

Input sequence	ANAKG		LUMAKG	
	Test I	Test II	Test I	Test II
What did the king had?	0	3	0	0
What stood in the garden?	0	2	0	1
Why was the king angry?	1	5	1	3
What were always counted?	2	2	2	2
What did the king order his gardener?	3	3	1	0
What was missing in the morning?	0	3	0	0

TABLE 7. RWP TO THE DESIRED OUTPUT, DISREGARDING INPUT SYMBOLS, IN BOTH TYPES OF MEMORIES.

Input sequence	ANAKG		LUMAKG	
	Test I	Test II	Test I	Test II
What did the king had?	1	0	1	1
What stood in the garden?	1	0	1	1/3
Why was the king angry?	7/10	1/10	7/10	3/10
What were always counted?	0	0	0	0
What did the king order his gardener?	0	0	3/8	1
What was missing in the morning?	1	0	1	1

The results applying RWP measure to the ANAKG and LUMAKG memory outputs from Tables 4 and 5 are shown in Table 7. These results also show that the performance of LUMAKG based semantic memory is equal to or better in all cases than ANAKG based semantic memory and its relative recall quality over ANAKG increases as the memory size increases.

D. Computational Complexity

The third type of tests was performed to determine computational complexity of LUMAKG memory in comparison to ANAKG memory. We tested the time needed to create the associative memory as a function of the number of objects. The results for a data set of all stories in Grimm's Fairy Tales are shown in Fig. 8-9. Fig. 8 shows the change in the number of unique objects (unique words) as the number of all objects (all words in all sentences) in the dataset increases. Neurons represent unique words in the neural knowledge graphs. As more input sequences are entered into the system, more mini-columns were previously introduced, and the rate of increase of unique objects slows down. Fig. 9 shows the learning time for both ANAKG and LUMAKG as a function of all objects in the data set. We see that computational cost for LUMAKG is between 30-40% higher than for ANAKG. Tests were performed on a general purpose laptop (i5-4300M CPU, 2.6 GHz, 8GB RAM).

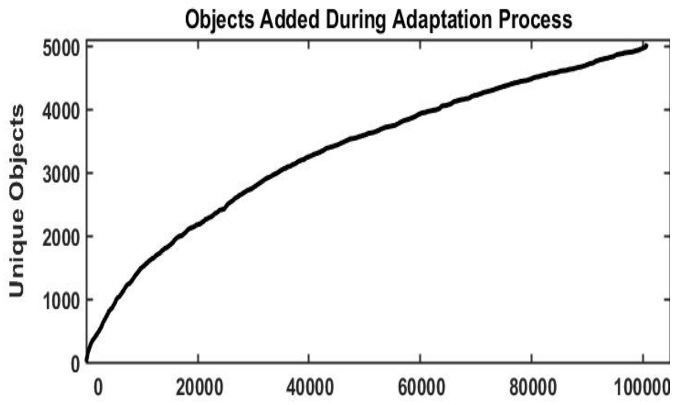

Fig. 8. Results for a large data set: number of neurons in ANAKG and mini-columns in LUMAKG as a function of the number of learned objects.

The number of neurons in LUMAKG memory is k times larger than in ANAKG memory, where k is the number of neurons in each mini-column (in our tests k=5). 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. They are just spread over the larger number of neurons. Although the training time is greater for LUMAKG than for ANAKG due to the need of finding predecessor and successor for each element of the training sequence, the quality of results points to better properties of LUMAKG graphs, which make them more suitable to develop short term associative memories.

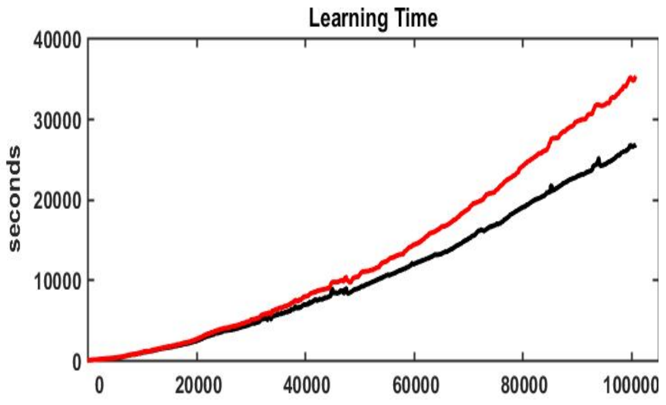

Fig. 9. Results for a large data set: learning times in ANAKG (lower curve) and LUMAKG (upper curve) as a function of the number of learned objects.

V. CONCLUSIONS

Presented in this paper LUMAKG memory supports continuous on-line learning, self-organization without supervised learning, context based predictions, and is capable of recognizing time-domain sequences correctly. LUMAKG shows better ability to recall sequences stored in the memories than ANAKG to which it was compared. Using 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 is to 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 a larger training data sets and a larger number of the test sequences to obtain a statistically sound assessment of the network properties. The effect of varying the number of neurons in minicolumns on performance will also be studied. ANAKG memories and its derivatives are new types of memories that are under intensive investigation. Their properties are explored with a final goal to use them as basic models for self-organization of the semantic memories.

ACKNOWLEDGMENT

This work was supported by a grant from the National Science Centre of Poland DEC2016/21/B/ST7/02220 and by AGH 11.11.120.612.

REFERENCES

[1] J.R. Bider, R.H. Desa, The neurobiology of semantic memory, Trends in Cognitive Sciences, vol. 15, issue 11, 2011, pp. 527-536.

[2] W. Kintsch, The role of knowledge in discourse comprehension. A construction-integration model. Psychological Review, vol. 95, 1988, pp. 163–182.

[3] S. Haykin, Neural networks. A comprehensive foundation, 1994.

[4] S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural computation, vol. 9, issue 8, no. 15, 1997, pp. 1735–1780.

[5] T. Mikolov, et al., Extensions of recurrent neural network language model. *IEEE Int. Conf. on Acoustics, Speech and Signal Processing* (ICASSP), Prague, pp. 5528-5531, 2011.

[6] J. Weston, S. Chopra, A. Bordes, Memory networks. 3rd Int. Conf. on Learning Representations (ICLR 2015), eprint arXiv.1410.3916v11 [cs.AI], 2015.

[7] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and D. Hal III, A neural network for factoid question answering over paragraphs. Conf. on Empirical Methods in Natural Language Processing (EMNLP) , 2014, pp. 633–644.

[8] A. Graves, G. Wayne, and I. Danihelka, Neural Turing Machines. arXiv preprint arXiv.1410.5401, 2014.

[9] A. Horzyk, How Does Generalization and Creativity Come into Being in Neural Associative Systems and How Does It Form Human-Like Knowledge? DOI. 10.1016/j.neucom.2014.04.046, Neurocomputing, vol. 144, 2014, pp. 238-257.

[10] A. Horzyk, J.A. Starzyk, and Basawaraj. Emergent creativity in declarative memories. 2016 IEEE Symposium Series on Computational Intelligence, DOI. 10.1109/SSCI.2016.7850029, Athens, Greece, 2016.

[11] J. Hawkins, S. Ahmad, and D. Dubinsky, Cortical learning algorithm and hierarchical temporal memory. http://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf, 2011.

[12] Y. Cui, C. Surpur, S. Ahmad, and J. Hawkins, Continuous online sequence learning with an unsupervised neural network model. arXiv.1512.05463 [cs.NE], 2015.

[13] H. Jaeger, Adaptive nonlinear system identification with echo state networks. In Advances in Neural Information Processing Systems 15 (NIPS 2002), pages 593–600. MIT Press, Cambridge, MA, 2003.

[14] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 1997, vol. 9, issue 8, 1997, pp. 1735-1780.

[15] W. Maass, Networks of spiking neurons: The third generation of neural network models, Neural Networks, Vol. 10, Issue 9, Elsevier, 1997, pp. 1659–1671.

[16] Izhikevich E.M. (2003) Simple Model of Spiking Neurons IEEE Transactions on Neural Networks, 14:1569- 1572.

[17] E.M. Izhikevich (2006), Polychronization: computation with spikes. Neural Computation vol. 18, no2, pp. 245–282.

[18] A. Horzyk, J.A. Starzyk, J. Graham, Integration of Semantic and Episodic Memories, IEEE Trans. on Neural Networks and Learning Systems, 2017, DOI: 10.1109/TNNLS.2017.2728203 (in press).

[19] https://www.cs.cmu.edu/~spok/grimmtmp/, last accessed 2017/08/07.

[20] S. K. Ray, S. Singh, and B. P. Joshi (2010), A semantic approach for question classification using WordNet and Wikipedia. Pattern Recognition Letters, 31(13), pp.1935-1943.

[21] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, 2003, July. Quantitative evaluation of passage retrieval algorithms for question answering. In Proc. of the 26th Annual Int. ACM SIGIR Conf. on Research and Development in Informaion Retrieval (pp. 41-47).

[22] S. Büttcher, C.L. Clarke, and G.V. Cormack, 2016. Information retrieval: Implementing and evaluating search engines. Mit Press.

[23] V.I. Levenshtein, Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics - Doklady, vol. 10, no.8, 1966, pp. 707-710.

[24] D. Radev, H. Qi, H. Wu and W. Fan, Evaluating web-based question answering systems, Proceedings of the 3rd Int. Conf. on Language Resources and Evaluation (2002).