2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE) | 978-1-6654-3897-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICECCE52056.2021.9514138

Exploring Neural Turing Machines Applicability in Neural-Symbolic Decision Support Systems

Alexander Demidovskij
Computer Science Department
Higher School of Economics
Nizhny Novgorod, Russia
ORCID: 0000-0003-3605-6332

***Abstract*—The task of building hybrid decision support systems that combine symbolic and connectionist approaches is actual and challenging. In particular, decision support systems operate with symbolic structures that describe the problem situation, stakeholders, assessment criteria, etc. Integrating connectionist approaches into certain parts of the decision-making process bring robustness, fixed response speed and ability to generalize. This paper examines Neural Turing Machines – a special case of Memory-Augmented Neural Networks – and demonstrates that such an architecture can be integrated into the Decision Support Systems. It was also shown that Neural Turing Machine can solve arithmetic sum task for numbers represented as binary vectors of length 10.**

***Keywords*—Neural Turing Machines, artificial neural networks, sub-symbolic computations, intelligent systems**

I. Introduction

One of the ways of building integrated neural-symbolic systems is ability to offload selected intellectual parts of these systems to neural networks by expressing symbolic algorithms as a neural network dynamic. A task of constructing neural-symbolic decision support systems is an actual and challenging goal [1]. Based on the criteria of the dominant component of Decision Support Systems (DSS) there are several categories of DSS: communications-driven and group DSS, data and document-driven DSS, knowledge-driven DSS, model-driven DSS, web-based and interorganizational DSS [2]. There are numerous discussions on the place that Artificial Neural Networks (ANN) might play in DSS [3], [4], [5].

ANN are often considered to play a role of universal processor or a quantitative model in model driven DSS. However, there are also works dedicated to teaching neural network to a specific problematic situation [6]. This model, though, includes training on a particular dataset, which is already difficult to come by in the field of decision-making. Aside from that, each task's neural network is unique, and each new Decision-Making situation needs retraining.

This paper considers a special aspect of the DSS – aggregation of assessments. Every decision-making task is characterized by the problem, the alternatives, criteria, experts' assessments etc. In order for the Decision Maker to choose the best alternative, all the assessments should be aggregated so that

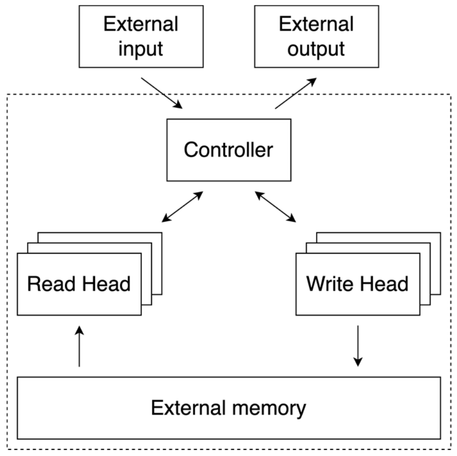

Fig. 1. Overall design of Neural Turing Machines

each alternative has a quantitative mark. Assessment's aggregation is a complicated process due to multiple reasons: assessments incompleteness, fuzzy nature of these assessments, for example linguistic ones, huge number of alternatives, conflicting criteria, expertise difference, etc. Assessment aggregation is a universal stage of numerous Decision-Making frameworks and methods: TOPSIS [7], ELECTRE [8], ML-LDM [9], etc. Therefore, construction of the neural-symbolic DSS could be started from expression of the fuzzy assessments' aggregation on the neural level. It is important to highlight that even when aggregation becomes fully represented by ANN, the overall DSS is still neural-symbolic, as problem description, the alternatives, criteria should be still described with symbols to keep the decision-making process interpretable and transparent for the Decision Maker and stakeholders. This paper examines Neural Turing Machines architecture capabilities to express arithmetic operations that are natural parts of the assessments aggregation process.

This paper is organized as follows. Section II contains short introduction to linguistic assessments aggregation mechanics. Section III reveals important details of Neural Turing Machines architecture. Section IV demonstrates application of Neural

The reported study was funded by RFBR, project number 19-37-90058.

Turing Machines to the arithmetic task. Conclusions are drawn in the final part of the paper.

II. Linguistic Assessments Aggregation

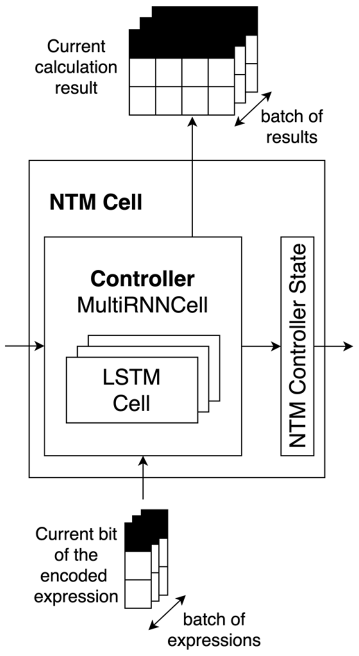

Fig. 2. Overall design of Neural Turing Machines

Modern methods of multi-attribute multi-level decision making use a traditional 2-tuple model as a basic building block [zhang]. An important feature of this model is ability to express both qualitative and quantitative assessments. The 2-tuple model is based on the concept of symbolic translation [10].

Definition 1. A 2-tuple structure includes a pair (s_i, α) where $s_i \in S = \{s_0, \dots, s_g\}$ – is a linguistic term (concept), α – a numeric value or a symbolic translation that shows a result of execution of membership function. It shows the distance to the closest concept $s_i \in S = \{s_0, \dots, s_g\}$ if a membership function does not result in an exact value (s_i).

Definition 2. Translation rule. Let $S = \{s_0, \dots, s_\tau\}$ be a linguistic scale, where $g = \tau + 1$ denotes granularity level of S. If $\beta \in [0,1]$ is a result of symbolic aggregation, then there is a way to recover a corresponding 2-tuple element:

$$\Delta_g = [0,1] \to S \times [-0.5, 0.5)$$
$$\Delta_g(\beta) = (s_i, \alpha) = \begin{cases} s_i, & i = round(\beta\tau) \\ \alpha = \beta\tau - i, & \alpha \in [-0.5, 0.5) \end{cases} \quad (1)$$

Definition 3. Reverse translation rule. Let $S = \{s_0, \dots, s_\tau\}$ be a linguistic scale, where $g = \tau + 1$ denotes granularity level of S. Let (s_i, α) be a 2-tuple element on a linguistic scale S, where $\alpha \in [-0.5, 0.5)$. Then there is a way to transform 2-tuple element to a numeric form of $\beta \in [0,1]$:

$$\Delta_g^{-1} = S \times [-0.5, 0.5) \to [0,1]$$
$$\Delta_g^{-1}(s_i, \alpha) = \frac{i + \alpha}{\tau} \quad (2)$$

There are multiple ways of aggregating assessments expressed in a form of 2-tuple, they are usually referred to as operators: MTWA (Multigranularity 2-tuple Weighted Averaging), MHTWA (Multigranularity Hesitant 2-tuple Weighted Averaging) [11], P2TLWA (Pythagorean 2-tuple Linguistic Weighted Averaging) [12] etc. One of them, MTWA, performs calculation of weighted average across a set of 2-tuple elements.

Definition 4. MTWA operator. Let (b_i, α_i) be a 2-tuple element on a linguistic scale S^{g_i}, $i = 1, 2, \dots, n$. Let $w = (w_1, w_2, \dots, w_n)$ be a given weighting vector where w_i denotes a weight for (b_i, α_i), $i = 1, 2, \dots, n$. Then the MTWA operator is defined by:

$$MTWA_{S^{g_k}}^{w}((b_1, \alpha_1), (b_2, \alpha_2), \dots, (b_n, \alpha_n)) = \Delta_{g_k}\left(\sum_{j=1}^{n} w_j \Delta_{g_j}^{-1}(b_j, \alpha_j)\right) \quad (3)$$

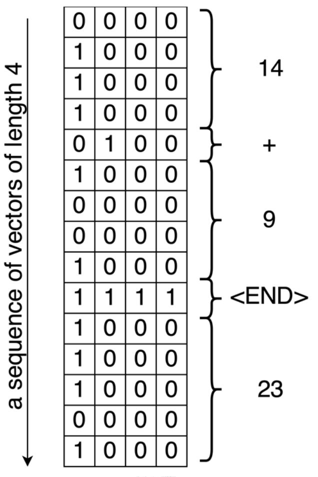

Fig. 3. Example encoding of arithmetic expression *14+9=23*

However, the 2-tuple model is a basic model, in recent years various methods that extend the original ideas of aggregating quantitative data were put forward: Hesitant Fuzzy Linguistic Term Sets (HFLTS) [13], Institutional 2-tuple [14], [15], hybrid models [16], etc. In general, each operator is associated with a number of arithmetic operations. In the discourse of current research, one of the directions towards building neural-symbolic DSS is expressing arithmetic operations in neural networks dynamic. This work continues series of works dedicated to elaborating neural-symbolic systems based on neural networks that do not require training [17]. Such neural networks operate on top of linguistic assessments encoded with Tensor Representations [18]. Current research aims at analyzing how capable are Neural Turing Machines to solve arithmetic operations over 10-bit numbers expressed as a sequence of vectors.

III. Neural Turing Machines

Neural Turing Machines were first introduced in 2014 [19], [20] and since that time gained huge popularity and adoption in the variety of tasks from simple algorithmic tasks to reinforcement learning [21], sequential recommendations [22], natural language transduction [23] etc.

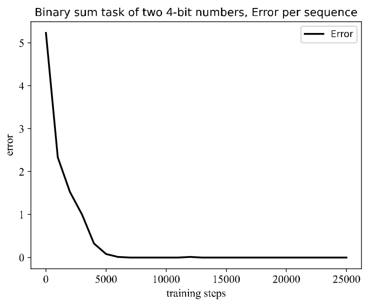

Fig. 4. Error per sequence, the task of binary sum of 4-bit numbers

Neural Turing Machines (NTM) are example of a Memory Augmented Neural Networks (MANN). The critical component of such an architecture is an additional memory, that is external to internal state of the neural network. In order to write to that memory and read from it, there are special abstractions called Heads. Finally, the Controller component performs the coordination of these heads in order to obtain the result. As NTM is a fully differentiable entity, NTMs can be successfully trained end-to-end. The theoretical architecture of NTMs is demonstrated in Fig. 1.

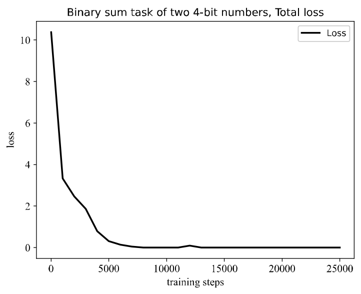

Fig. 5. Loss, the task of binary sum of 4-bit numbers

Implementation aspects of NTM are well described [24], [25] and mostly cover various approaches to making the training more stable by introducing gradient clipping mechanism. One important note is the error function that we use as a criterion to evaluate model training quality (2).

$$\delta_{i \geq 0.5} = \begin{cases} 1, & i \geq 0.5 \\ 0, & i < 0.5 \end{cases} \qquad (1)$$

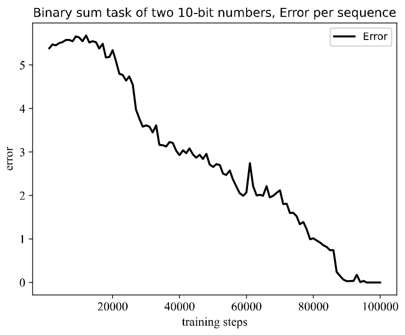

Fig. 6. Error per sequence, the task of binary sum of 10-bit numbers

$$e_{per\_sequence} = \sum_{b=1}^{B} \sum_{n=1}^{N+1} \sum_{m=1}^{M} |\delta_{X_{b,n,m} \geq 0.5} - Y_{b,n,m}| \qquad (2)$$

, where B denotes the batch size of the training data fed into the network, N is the number of bits per individual number and M is the length of vector containing single bit of the expression.

IV. Evaluation of the Neural Turing Machine Solving Binary Sum Task for 10-bit Numbers

However, from the practical implementation standpoint, NTM architecture is wrapped into the additional abstraction called NTM Cell that has a Controller network, access to external memory and necessary heads to operate with that memory. More importantly, there is a known ability of NTMs to generalize for the task they are designed for. In particular, NTM might be trained on fixed-length sequences of vectors and once it is trained, the network can demonstrate the task being solved for sequences of bigger length. A network designer might want to use this generalization ability of the network and for that the NTM Cell abstraction is required as it allows to stack as many instances of NTMs as needed or design the most generic solution that would be a dynamic Recurrent Neural Network (d-RNN) that dynamically unrolls input sequence.

NTM is trained in a supervised manner. During the training phase NTM expects a batch of vector sequences as the input and the batch of labels that is another sequence of vectors that stand for the expected network output. During the inference phase NTM expects a batch of vector sequences that means that NTM can perform the same task over multiple sequences simultaneously. There are numerous tasks that the NTM was proved to solve, such as Copy, Repeat, Associative Recall, Binary Addition, Binary Multiplication, N-Gram of the Dynamic N-Grams Task, Priority Sort Task. In this work, we consider Binary Addition task for numbers with an adequate bit length, as 4-bits in the original paper is not enough for industrial tasks. Overall scheme of feeding data into the neural network is demonstrated in Fig. 2.

In this paper, we try to solve the arithmetic task of making a sum of two numbers. The whole arithmetic expression is encoded as a matrix of a certain format. In particular, each number is represented as a matrix with N rows and M columns, where N stand for the number of bits we are using to encode the number. For number encoding we are using the Little-endian format. In order to help the network distinguish between bits encoding the number and bits that encode arithmetic operation of sum and marker that means end of number, we use several "channels" in the data. In particular, the first column in matrix contains bits of numbers, the second contains bits for arithmetic operations, marker of the expression end is represented as a vector of ones of length M. Example arithmetic expression is demonstrated in Fig. 3.

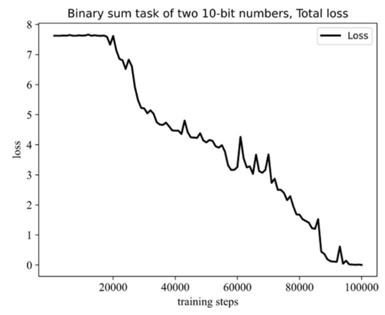

Fig. 7. Loss, the task of binary sum of 10-bit numbers

As we can see from the Fig. 3, only 4 bits are used to represent input numbers. However, 5 bits are used to store the results of the sum in order to keep the potential overflow. Using 4 bits in such arithmetic expressions allows us to express numbers in the range from 0 to 15. From our point of view, it is important to try using NTM to learn to make sum of numbers of bigger precision, for example 10 bits. Obviously, that might dramatically increase the range of numbers we can sum, namely from 0 to 1023.

We have conducted experiments with bit strings of length 4 and 10. In particular, we trained an NTM on generated dataset, where each entity was represented according to the certain format (Fig. 3) and packed into batches of size 32.

In case of 4-bit numbers, NTM was able to show zero error (Fig. 4) and significantly minimize loss (Fig. 5) after 7000 steps. In case of 10-bit numbers, NTM was also able to show zero error (Fig. 6) and significantly minimize loss (Fig. 7) after 112000 steps.

All experiments were made on the following configuration: Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz (frequency was not fixed) with 8152 Mb RAM. All results can be reproduced with an open-source project (https://github.com/demid5111/NeuralTuringMachine). The project is based on top of existing TensorFlow implementation of Neural Turing Machine [24].

V. Conclusion

This work is dedicated to exploring building hybrid decision support systems by combining symbolic and connectionist approaches. We examined Neural Turing Machines – a special case of Memory-Augmented Neural Networks – and demonstrated that such an architecture can be integrated into the Decision Support Systems. It was also shown that Neural Turing Machine can solve arithmetic sum task for numbers represented as binary vectors of length 10. It is suggested to analyze Neural Turing Machine capability to learn and generalize the algorithm of aggregation operators, such as MTWA.

References

[1] Kasabov, Nikola. "Evolving connectionist-based decision support systems." In Applied Decision Support with Soft Computing, pp. 86-98. Springer, Berlin, Heidelberg, 2003.

[2] Power, Daniel J. Decision support systems: concepts and resources for managers. Greenwood Publishing Group, 2002.

[3] Delen, Dursun, and Ramesh Sharda. "Artificial neural networks in decision support systems." In *Handbook on Decision Support Systems 1*, pp. 557-580. Springer, Berlin, Heidelberg, 2008.

[4] Power, Daniel J., and Ramesh Sharda. "Model-driven decision support systems: Concepts and research directions." *Decision support systems* 43, no. 3 (2007): 1044-1061.

[5] Matzkevich, Izhar, and Bruce Abramson. "Decision analytic networks in artificial intelligence." *Management Science* 41, no. 1 (1995): 1-22.

[6] Golmohammadi D. Neural network application for fuzzy multi-criteria decision making problems. International Journal of Production Economics, vol. 131, i. 2, pp. 490-504.

[7] Yoon K., C.L. Hwang. TOPSIS (technique for order preference by similarity to ideal solution)–a multiple attribute decision making. In Multiple attribute decision making–methods and applications, 1981

[8] Figueira José, Vincent Mousseau, Bernard Roy. ELECTRE methods. In Multiple criteria decision analysis: State of the art surveys, 2005, pp. 133-153.

[9] Demidovskij A.V., Babkin E.A. Developing a distributed linguistic decision making system. Business-informatics, vol. 13, i. 1.

[10] Herrera F., Martínez L. A 2-tuple fuzzy linguistic representation model for computing with words. IEEE Transactions on fuzzy systems, vol. 8, i. 6, pp. 746-752.

[11] Wei C., Liao H. A multigranularity linguistic group decision-making method based on hesitant 2-tuple sets. International Journal of Intelligent Systems, vol. 31, i. 6, pp. 612-634.

[12] Wei G., Lu M., Alsaadi F.E., Hayat T., Alsaedi A. Pythagorean 2-tuple linguistic aggregation operators in multiple attribute decision making. Journal of Intelligent & Fuzzy Systems, vol. 33, i. 2, pp. 1129-1142.

[13] Rodriguez R.M., Martinez L., Herrera F. Hesitant fuzzy linguistic term sets for decision making. IEEE Transactions on fuzzy systems, vol. 20, i. 1, pp. 109-119.

[14] Liu P., Chen S. M. Multiattribute group decision making based on intuitionistic 2-tuple linguistic information. Information Sciences, vol. 430, pp. 599-619.

[15] Wei G., Alsaadi F.E., Hayat, T., Alsaedi A. Picture 2-tuple linguistic aggregation operators in multiple attribute decision making. Soft Computing, vol. 22, i. 3, pp. 989-1002.

[16] Wang J. H., Hao J. A new version of 2-tuple fuzzy linguistic representation model for computing with words. IEEE transactions on fuzzy systems, vol. 14, i. 3, pp. 435-445.

[17] Demidovskij, Alexander, and Eduard Babkin. "Designing a Neural Network Primitive for Conditional Structural Transformations." In *Russian Conference on Artificial Intelligence*, pp. 117-133. Springer, Cham, 2020.

[18] Demidovskij, A. and Babkin, E., 2020, December. Designing arithmetic neural primitive for sub-symbolic aggregation of linguistic assessments. In *Journal of Physics: Conference Series* (Vol. 1680, No. 1, p. 012007). IOP Publishing.

[19] Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." *arXiv preprint arXiv:1410.5401* (2014).

[20] Graves, Alex, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo et al. "Hybrid computing using a neural network with dynamic external memory." *Nature* 538, no. 7626 (2016): 471-476.

[21] Zaremba, Wojciech, and Ilya Sutskever. "Reinforcement learning neural turing machines-revised." *arXiv preprint arXiv:1505.00521* (2015).

[22] Chen, Xu, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. "Sequential recommendation with user memory networks." In Proceedings of the eleventh ACM international conference on web search and data mining, pp. 108-116. 2018.

[23] Grefenstette, Edward, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. "Learning to transduce with unbounded memory." *arXiv preprint arXiv:1506.02516* (2015).

[24] Collier, Mark, and Joeran Beel. "Implementing neural turing machines." In *International Conference on Artificial Neural Networks*, pp. 94-104. Springer, Cham, 2018.

[25] Castellini, Jacopo. "Learning Numeracy: Binary Arithmetic with Neural Turing Machines." *arXiv preprint arXiv:1904.02478* (2019).