Neural Multigranular 2-tuple Average Operator in Neural-Symbolic Decision Support Systems

Alexander Demidovskij(✉) and Eduard Babkin

National Research University Higher School of Economics, Nizhny Novgorod, Russia
{ademidovskij,eababkin}@hse.ru

Abstract. Construction of integrated neural-symbolic systems is an actual and challenging task. Such hybrid systems combine advantages of connectionist and symbolic approaches. In particular, neural-symbolic systems are characterized by robust learning and distributed neural computations. At the same time, they can be interpreted, described and analyzed in logical terms. Especially high interpretability is important for Decision Support Systems that operate with symbolic structures describing the problem situation, stakeholders, assessment criteria and where the reasoning process should be transparent for the Decision Maker. Such requirements bring the task of designing integrated neural-symbolic Decision Support Systems to the higher level of complexity. This paper examines underlying algorithms of a specific part of Decision Support Systems that is aggregation of experts' assessments to make the choice among alternative solutions of the given problem. Such fuzzy and uncertain assessments are often represented as 2-tuple model or its derivatives and algorithms that aggregate them are often called operator. We demonstrate that the simple aggregation operator can be expressed in a fully connectionist form with the help of Neural Turing Machine architecture. This result sheds new light on the way principles of symbolic computation can be implemented by connectionist mechanisms.

Keywords: Neural Turing Machines · Artificial Neural Networks · Sub-symbolic computations · Intelligent systems

1 Introduction

For a long period of time, the fields of Artificial Intelligence (AI) and Decision Analysis (DA) evolved concurrently, despite the fact that both fields research formal models of human knowledge and expertise [1]. Decision Support Systems (DSS) were developed as a tool for automating the compilation and review of expert and stakeholder opinions in order to assist the Decision Maker (DM) in making the decision. Thus, DSS should be viewed as a utility structure that contains certain rules and algorithms for advising the DM. Simultaneously, Expert Systems were proposed in the field of Artificial Intelligence to collect large volumes of domain data from a large number of domain experts, encode this information, and apply this knowledge to any instance of a domain

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

S. Kovalev et al. (Eds.): IITI 2021, LNNS 330, pp. 350–359, 2022.
https://doi.org/10.1007/978-3-030-87178-9_35

problem that occurs [2]. Both methods, however, suffer from an inability to deal with knowledge complexity and expert confusion. Numerous factors should be considered, including the collection of ambiguous expert assessments of alternative options, the use of incomplete data, and so on. Such complexities have established a common ground for inter-disciplinary communication, and various approaches have been developed to resolve the aforementioned issues. However, one of the central issues is how to represent knowledge in a way that is computationally efficient.

It is clear that human cognition is capable of dealing with knowledge ambiguity and incompleteness and performs admirably when confronted with extremely challenging puzzles. This fact casts new light on the fields of Decision Analysis and Artificial Intelligence. Numerous researchers are motivated by the fact that human cognition makes use of an incredibly large neural network as a computational engine and propose that the human cognitive system makes use of a distributed representation of information and processes it in a dynamic and meaningful manner. This neural-based approach to information representation and processing is referred to as the sub-symbolic or connectionist approach. Simultaneously, another school of thought advocates for seeing human cognition as a symbolic manipulation method [3], interpreted as "a species of computation carried out in a specific type of biological system" [4]. Within this approach symbols appear as inputs to such systems and then they are converted into other symbols using predefined rules and instruction sets. This approach is referred to as a symbolic approach.

As can be shown, the problems confronting the area of human cognition research and decision-making are strikingly similar in terms of information representation and processing. Although there are distinct methods, symbolic and sub-symbolic, there is a clear preference for integrated solutions. The primary reason for this is that each of these methods has distinct advantages and disadvantages that prevent any of them from being the magic bullet.

Sub-symbolic approaches are distributed by nature and have high efficiency and robustness. Additionally, they incorporate a critical component of learning, allowing sub-symbolic methods to be "trained" for a specific task and continuously modified as new information becomes available. At the same time, such approaches are difficult to understand, although an emerging field called Explainable AI (xAI) [5] is devoted to developing methods for deriving meaning from complex computational models. Another disadvantage of sub-symbolic methods is their inability to articulate complex relationships due to the limitations of current methods for distributed information representation [3, 4]. Recent advancements in the field have enabled the resolution of several representational issues associated with sub-symbolic architectures: the introduction of Tensor Product Representations [6] capable of representing recursion in symbolic structures; further distributed representation size enhancements through Holographic Reduced Representations (HRR) [7] and other methods of expressing symbolic computations. Typically, this class of methods is referred to as Vector Symbolic Architectures (VSA) [8, 9].

On the other hand, although symbolic approaches are extremely interpretable, they are by definition sequential. Significantly, symbolic approaches presuppose the creation of processing rules and instruction sets, which has two implications. Firstly, the information encoded in a symbolic system by a programmer represents their own expertise and worldview, making the system biased. This issue has been identified as a symbol grounding issue [10]. Secondly, the information stored in the symbolic DSS degrades rapidly.

As a result, we can see how an integrated symbolic and sub-symbolic approach forms the basis for systems such as ACT-R [11], CLARION [12], and SS-RICS [13, 14]. Psychophysiological plausibility was established by demonstrating the presence of interconnected architectures in the human nervous system and various biological architectures. Specifically, "one may consider the two methods as two ends of a single continuum: $<\dots>$ sub-symbolic systems accept input and move it along to more symbolic systems." [3, p. 850]. Construction of neural-symbolic integrated systems is a current challenge in the area of Decision Analysis [15, 16]. Such systems will combine sub-symbolic reasoning and computation at the connectionist or neural level with symbolic reasoning and computation at the symbolic level.

However, ongoing controversies exist about what it means for DSS to become neural-symbolic or integrated [1, 16, 17]. In particular, what DSS components can be delegated to the sub-symbolic level and what function Artificial Neural Networks (ANN) can play in DSS. There are several types of DSS based on the criteria for the dominant component: DSSs that are communication- and group-driven, data- and document-driven, knowledge-driven, model-driven, web-based, and interorganizational [18]. According to [16], ANN can be used effectively for forecasting using historical data and can also be called quantitative models. In other words, ANNs can be used in both data-driven and model-driven DSS. However, applying ANN to data-driven DSS often requires teaching the neural network to solve a particular problem [19]. This requires training on a specific dataset, which is already scarce in the decision-making sector. Apart from that, each job requires a specific neural network, and each new Decision-Making scenario requires retraining.

Simultaneously, seeing ANN as a quantitative model with learning capabilities enables a DSS architect to articulate such low-level operations and symbolic transformations at the sub-symbolic level. As a consequence of the ANN's architecture peculiarities, these transformations are performed with marginal information loss while becoming more performant and robust.

This article focuses on a special aspect of the DSS: assessment aggregation. Any decision-making task is described by the problem, the solutions, the requirements, and expert evaluations, among other things. To assist the Decision Maker in selecting the best alternative, all evaluations should be aggregated into a quantitative mark for each alternative. Aggregating evaluations is a challenging task for a variety of reasons, including the assessments' incompleteness, the fuzzy nature of certain assessments, such as linguistic assessments, the vast number of alternatives, conflicting standards, and gaps in expert's competence. Aggregation of assessments is a common stage in a variety of decision-making methods, including TOPSIS [20], ELECTRE [21], and ML-LDM [22]. Thus, construction of the neural-symbolic DSS may begin with the expression of the aggregation of fuzzy assessments at the neural level. Additionally, when aggregation is fully articulated using ANN, the overall DSS remains neural-symbolic, as problem descriptions, alternatives, and parameters should all be described using symbols to keep the DM and stakeholders informed and involved in the decision-making process. This article explores the capabilities of Neural Turing Machines' architecture to express linguistic assessments aggregation operator.

This article is organized as follows. Section 2 includes a brief introduction to the mechanics of linguistic assessments aggregation. Section 3 delves into the design of Neural Turing Machines. Section 4 illustrates the use of Neural Turing Machines to express the assessments aggregation operator. The final part of the paper contains the conclusions.

2 Linguistic Assessments Aggregation with Multigranular 2-tuple Averaging Operator

Modern methods of multi-attribute multilevel decision making use a traditional 2-tuple model as a basic building block [23]. 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 [24].

Definition 1. A 2-tuple structure includes a pair (s_i, α) where $s_i \in S = \{s_0, \ldots, s_g\}$ – 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 $s_i \in S = \{s_0, \ldots, s_g\}$ if a membership function does not result in an exact value (s_i).

Definition 2. Translation rule. Let $S = \{s_0, \ldots, s_g\}$ 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:

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

Definition 3. Reverse translation rule. Let $S = \{s_0, \ldots, s_\tau\}$ 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 $\beta \in [0, 1]$:

$$\begin{gathered}\Delta_g^{-1} = S \times [-0.5,\ 0.5) \to [0, 1] \\ \Delta_g^{-1}(s_i,\ \alpha) = \tfrac{i+\alpha}{\tau}\end{gathered} \tag{2}$$

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) [25], P2TLWA (Pythagorean 2-tuple Linguistic Weighted Averaging) [26] etc. One of them, MTA (Multigranularity 2-tuple Averaging), performs calculation of weighted average across a set of 2-tuple elements.

Definition 4. MTA operator. Let (b_i, α_i) be a 2-tuple element on a linguistic scale S^{g_i}, $i = 1, 2, \ldots, n$. Then the MTA operator is defined by:

$$MTA((b_1, \alpha_1), (b_2, \alpha_2), \ldots, (b_n, \alpha_n)) = \Delta_{g_k}\left(\sum_{j=1}^{n} \frac{1}{n} \Delta_{g_j}^{-1}(b_j, \alpha_j)\right) \tag{3}$$

Although the 2-tuple model is a fundamental model, numerous methods for aggregating fuzzy assessments have been proposed in recent years: Hesitant Fuzzy Linguistic Term Sets (HFLTS) [27], Institutional 2-tuple [28, 29], hybrid models [30], and so on. Each operator is typically associated with a set of arithmetic operations.

According to current research, one method for constructing neural-symbolic DSS is to express arithmetic operations dynamically in neural networks. This article is part of a series on neural-symbolic architecture of DSS [31]. However, this paper takes a different approach to DSS design than [31], which in turn focused on proposing such neural network architectures that do not require training and that run on top of linguistic assessments encoded with Tensor Representations [32]. The current research examines how capable Neural Turing Machines to express an MTA operator.

3 Neural Turing Machines

3.1 Architecture

Neural Turing Machines were first introduced in 2014 [33, 34] and since that time gained huge popularity and adoption in the variety of tasks from simple algorithmic tasks to reinforcement learning [35], sequential recommendations [36], natural language transduction [37] etc.

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.

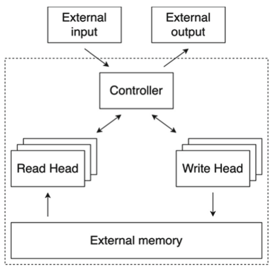

Fig. 1. Overall design of Neural Turing Machines

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 expression of MTA operator for numbers expressed with 4, 6, 8, 10 bits. Overall scheme of feeding data into the neural network is demonstrated in Fig. 2.

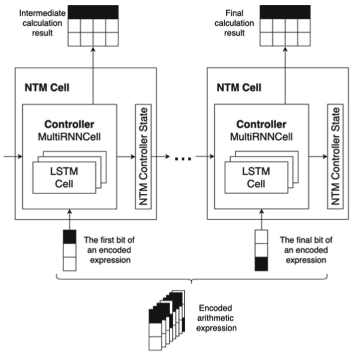

Fig. 2. Feeding NTM with an encoded data for MTA

3.2 Implementation Aspects

Implementation aspects of NTM are well described [38, 39] 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 (5).

$$\delta_{i \geq 0.5} = \begin{cases} 1, i \geq 0.5 \\ 0, i < 0.5 \end{cases} \tag{4}$$

$$e_{per\_sequence} = \sum_{b=1}^{B} \sum_{n=1}^{N+1} \sum_{m=1}^{M} |\delta_{X_{b,n,m} \geq 0.5} - Y_{b,n,m}|, \tag{5}$$

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.

In practice, NTM architecture is wrapped in an additional abstraction called NTM Cell, which includes a Controller network, access to external memory, and the required heads to operate with that memory. More importantly, NTMs are able to generalize. In particular, NTM can be trained on fixed-length vector sequences and once trained, the network can show that the same task is solved for longer sequences. A network designer may wish to leverage the network's generalization capability, for which the NTM Cell abstraction is necessary, as it enables the stacking of as many NTM instances as required, or design the most generic solution possible, which is a dynamic Recurrent Neural Network (d-RNN) that dynamically unrolls the input series. All of the findings presented below are reproducible within an open-source project [40]. The project is built on top of the current TensorFlow Neural Turing Machine implementation [24].

4 Evaluating Neural Turing Machines as Neural MTA Operator

We express an MTA operator that aggregates assessments from three experts; each assessment is converted to numeric form and represented as a binary string [39]. We experimented with various binary string lengths: 4, 6, 8, 10 bits.

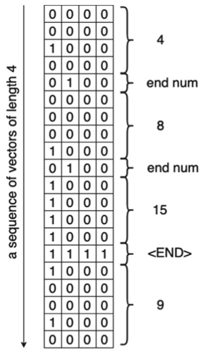

Fig. 3. Encoded expression for MTA

The entire arithmetic expression is encoded in the form of a matrix with a specific format. Each number is expressed mathematically as a matrix with N rows and M columns, where N denotes the number of bits used to encode the number. We are using the Little-Endian format for number encoding. We use many "channels" in the data to assist the network in distinguishing between bits that encode the number and bits that encode markers. The first column of the matrix contains bits of data, the second column contains bits for markers, and the end marker is represented by a vector of ones of length M. Figure 3 illustrates an arithmetic expression. For training purposes, we created a dataset in which each entity was represented using a specific format (Fig. 3) and compressed into batches with 32 arithmetic expressions in each.

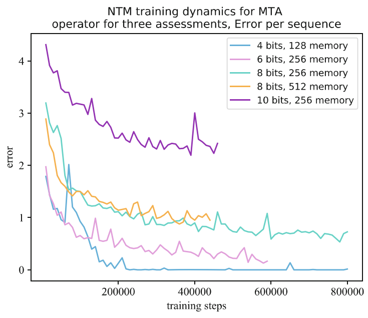

Fig. 4. NTM training dynamics. Error per sequence

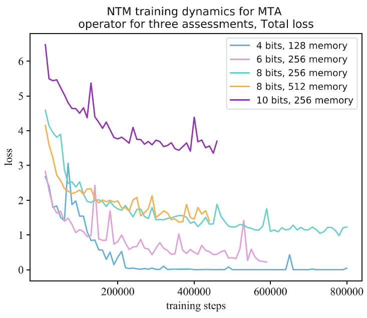

Fig. 5. NTM trainings dynamics. Total loss

NTM was able to achieve zero error only in case of 4-bit numbers (Fig. 4). For other bit length there is an error varying from 0.26 for 6 bits to 2.42 for 10-bit numbers. At the same time, NTM was able to substantially minimize loss in all cases (Fig. 5). All experiments were made on various configurations: Intel(R) Xeon(R) CPU E5–2650 v4 @ 2.20 GHz, AMD EPYC 7282 16-Core Processor @ 2.79 GHz, Intel(R) Xeon(R) Gold 6140 CPU @ 2.30 GHz with not fixed frequency and 60GB RAM.

5 Conclusions

The aim of this research is to investigate various options for integrating symbolic and connectionist approaches in order to create hybrid DSS. We examined Neural Turing Machines, a subtype of Memory-Augmented Neural Network, and demonstrated their integration into Decision Support Systems. Additionally, it was shown that the MTA operator can be entirely represented on a neural level using a trained Neural Turing Machine. It is suggested to investigate the capabilities of Neural Turing Machines to learn and generalize another aggregation algorithm, the MTWA operator, since it is the simplest operator used in development of Linguistic DSS systems. Developing such solutions, in our opinion, provides the fundamental building blocks for new hybrid neural-symbolic Decision Support Systems.

References

1. Matzkevich, I., Abramson, B.: Decision analytic networks in artificial intelligence. Manage. Sci. **41**(1), 1–22 (1995)
2. Silverman, B.G.: Unifying expert systems and the decision sciences. Oper. Res. **42**(3), 393–413 (1994)
3. Kelley, T.D.: Symbolic and sub-symbolic representations in computational models of human cognition: what can be learned from biology? Theory Psychol. **13**(6), 847–860 (2003)
4. Pylyshyn, Z.W.: Computing in Cognitive Science. University of Western Ontario, Centre for Cognitive Science, London (1988)
5. Turek, M.: Explainable Artificial Intelligence (XAI) (2018). https://www.darpa.mil/program/explainable- artificial- intelligence. Accessed 27 June 2020
6. Smolensky, P., Legendre, G.: The Harmonic Mind: From Neural Computation to Optimality-Theoretic Grammar (Cognitive Architecture), vol. 1. MIT press, Cambridge (2006)
7. Plate, T.A.: Holographic reduced representations. IEEE Trans. Neural Netw. **6**(3), 623–641 (1995)
8. Kanerva, P.: Hyperdimensional computing: an introduction to computing in distributed representation with high-dimensional random vectors. Cogn. Comput. **1**(2), 139–159 (2009)
9. Schlegel, K., Neubert, P., Protzel, P.: A comparison of vector symbolic architectures. arXiv preprint arXiv:2001.11797 (2020)
10. Harnad, S.: The symbol grounding problem. Physica D **42**(1–3), 335–346 (1990)
11. Anderson, J.R., Lebiere, C.J.: The Atomic Components of Thought. Psychology Press, Hove (2014)
12. Sun, R., Peterson, T.: A hybrid agent architecture for reactive sequential decision making. Connectionist symbolic integration: From unified to hybrid approaches, pp.113–138 (1997)

13. Avery, E., Kelley, T.D., Davani, D.: May. Using cognitive architectures to improve robot control: Integrating production systems, semantic networks, and sub-symbolic processing. In: 15th Annual Conference on Behavioral Representation in Modeling and Simulation (BRIMS) (2006)
14. Kelley, T., Avery, E., Long, L., Dimperio, E.: A hybrid symbolic and sub-symbolic intelligent system for mobile robots. In: AIAA Infotech@ Aerospace Conference and AIAA Unmanned... Unlimited Conference, p. 1976 (2009)
15. Parvar, J., Lowe, D., Emsley, M., Duff, R.: Neural networks as a decision support system for the decision to bid process. In: Proceedings ARCOM Conference, vol. 1, pp. 209–217 (2000)
16. Delen, D., Sharda, R.: Artificial neural networks in decision support systems. In: Burstein, F., Holsapple, C.W. (eds.) Handbook on Decision Support Systems 1, pp. 557–580. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-48713-5_26
17. Power, D.J., Sharda, R.: Model-driven decision support systems: concepts and research directions. Decis. Support Syst. **43**(3), 1044–1061 (2007)
18. Power, D.J.: Decision Support Systems: Concepts and Resources for Managers. Greenwood Publishing Group, Westport (2002)
19. Golmohammadi, D.: Neural network application for fuzzy multi-criteria decision making problems. Int. J. Prod. Econ. **131**(2), 490–504 (2011)
20. Yoon, K., Hwang, C.L.: TOPSIS (Technique for Order Preference by Similarity to Ideal Solution)–A Multiple Attribute Decision Making, W: Multiple Attribute Decision Making–Methods and Applications, A State-of-the-at Survey. Springer, Heidelberg (1981).https://doi.org/10.1007/978-3-642-48318-9
21. Figueira, J.R., Mousseau, V., Roy, B.: ELECTRE methods. In: Greco, S., Ehrgott, M., Figueira, J.R. (eds.) Multiple criteria decision analysis. ISORMS, vol. 233, pp. 155–185. Springer, New York (2016). https://doi.org/10.1007/978-1-4939-3094-4_5
22. Demidovskij, A.V., Babkin, E.A.: Developing a distributed linguistic decision making system. Bus. Inf. **13**(1), 18–32 (2019)
23. Zhang, H., Wu, Y., Gao, J., Xu, C.: A method for multi-criteria group decision making with 2-tuple linguistic information based on cloud model. Information **8**(2), 54 (2017)
24. Herrera, F., Martínez, L.: A 2-tuple fuzzy linguistic representation model for computing with words. IEEE Trans. Fuzzy Syst. **8**(6), 746–752 (2000)
25. Wei, C., Liao, H.: A multigranularity linguistic group decision-making method based on hesitant 2-tuple sets. Int. J. Intell. Syst. **31**(6), 612–634 (2016)
26. Wei, G., Lu, M., Alsaadi, F.E., Hayat, T., Alsaedi, A.: Pythagorean 2-tuple linguistic aggregation operators in multiple attribute decision making. J. Intell. Fuzzy Syst. **33**(2), 1129–1142 (2017)
27. Rodriguez, R.M., Martinez, L., Herrera, F.: Hesitant fuzzy linguistic term sets for decision making. IEEE Trans. Fuzzy Syst. **20**(1), 109–119 (2011)
28. Liu, P., Chen, S.M.: Multiattribute group decision making based on intuitionistic 2-tuple linguistic information. Inf. Sci. **430**, 599–619 (2018)
29. Wei, G., Alsaadi, F.E., Hayat, T., Alsaedi, A.: Picture 2-tuple linguistic aggregation operators in multiple attribute decision making. Soft. Comput. **22**(3), 989–1002 (2016). https://doi.org/10.1007/s00500-016-2403-8
30. Wang, J.H., Hao, J.: A new version of 2-tuple fuzzy linguistic representation model for computing with words. IEEE Trans. Fuzzy Syst. **14**(3), 435–445 (2006)
31. Demidovskij, A., Babkin, E.: Designing a neural network primitive for conditional structural transformations. In: Kuznetsov, S.O., Panov, A.I., Yakovlev, K.S. (eds.) Artificial Intelligence: 18th Russian Conference, RCAI 2020, Moscow, Russia, October 10–16, 2020, Proceedings, pp. 117–133. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-59535-7_9

32. Demidovskij, A., Babkin, E.: Designing arithmetic neural primitive for sub-symbolic aggregation of linguistic assessments. In: Journal of Physics: Conference Series, vol. 1680, No. 1. IOP Publishing, December (2020)
33. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines. arXiv preprint arXiv:1410.5401 (2014)
34. Graves, A., et al.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471–476 (2016)
35. Zaremba, W., Sutskever, I.: Reinforcement learning neural turing machines-revised. arXiv preprint arXiv:1505.00521 (2015)
36. Chen, X., et al.: Sequential recommendation with user memory networks. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 108–116, February 2018
37. Grefenstette, E., Hermann, K.M., Suleyman, M., Blunsom, P.: Learning to transduce with unbounded memory. arXiv preprint arXiv:1506.02516 (2015)
38. Collier, M., Beel, J.: Implementing neural turing machines. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11141, pp. 94–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01424-7_10
39. Castellini, J.: Learning numeracy: binary arithmetic with neural turing machines. arXiv preprint arXiv:1904.02478 (2019)
40. Demidovskij, A.: Neural turing machine: open-source project. https://github.com/demid5111/NeuralTuringMachine Accessed 7 Apr 2021