Decoding Explicit Memory: A Toy Task for State Maintenance

Chaichan Poonperm
School of Engineering and Technology
Asian Institute of Technology
Klong Luang,
Pathum Thani 12120
Email: st121389@ait.asia

Akraradet Sinsamersuk
School of Engineering and Technology
Asian Institute of Technology
Klong Luang,
Pathum Thani 12120
Email: st121413@ait.asia

***Abstract*—Neural networks with spatial specialization make a huge success in computer vision and the performance is moving towards that of human. State-of-the-arts are rapidly published with real time performance. However, natural language processing is still progressing with a constant pace. Improvement ranges from recurrence to attention mechanism and the performance begins to rise significantly. There have been attempts to reuse one of the state of the arts to pre-train a large language model, with specialized tasks to performs feat in natural language processing domain. From millions to billions parameters, such model started to show an impressive result such as language modeling that bears a high degree of resemblance to human skills. However, it is not clear how the models manipulate the data. Here, we take an advantage from explicit memory in which the information used to perform a task is forced to be embedded inside the memory with. We make the calendar appointment task in which a calendar state changes when fed a new appointment from a user. Framed as a classification task with the purpose of reconstruction, we show empirically that the memory is able to store the information and be maintained as needed in the task.**

I. Introduction

Advance in neural networks makes it possible to overcome complex tasks requiring complex pattern recognition, especially computer vision. With specialization in sparse weight and interaction, Convolutional Neural Networks (CNNs) have given breakthrough in image processing tasks such as image classification [2], [3], with the performance competitive to that of humans. In parallel, there has also been attempts to deal natural language with neural networks. Recurrent Neural Networks (RNNs) enable better generalization than ordinary neural networks but it is still more than far to success even with gating mechanism. RNNs are then split to encoder-decoder architecture, and are further equipped with attention mechanism [4], [5] to grab important information from hidden states for the prediction of the output directly rather than lossy summarization over time. The Transformer [6] discards the sequential part in the encoder and relies solely on the attention mechanism which has shown to be superior to the previous RNN-based encoder-decoder architectures in machine translation task [4], [5], [7]. Moreover, the output from the attention mechanism evinces meaningful relationship between input that can be analyzed further. To this point, there are two ways in which researchers advance the models. First, the Transformer architecture is used to train a large language model and used for fine-tuning downstream tasks. Fine-tuning a pre-trained model based on the encoder of the Transformer shows state-of-the-art performance in several large-scale tasks [8]. A large pre-trained language model [9] shows impressive results in language modeling given various kinds of contexts. With more advancement, the size of the models tend grow larger from millions of parameters to billions [10], [11], [9]. Second, embedding memory to neural networks or neural networks with explicit memory is another way to cope with natural language. Neural Turing Machine [12] and Memory Networks [13] are pioneers in the augmentation. The former uses memory that is an additional matrix coupled to the neural networks with soft read and write operations, which has the same concept of the Turing Machine with a slightly different way of access. The latter provides a framework for the neural networks to interact with a separate memory. An instantiation has the same concept of the information retrieval when selecting appropriate memory locations to produce an output. End-to-end Memory Network [14] regard the embedding vectors of the input as memory locations which is compared to that of query in order to produce answer. Differentiable Neural Computer (DNC) [15], a successor of [12], has a more sophisticate way of managing memory, enabling more effective read and write. A set of GRUs can be arranged to convert from input to hidden states, output to hidden states, process the similarity between input and output, and produce answer [16]. Arrange hidden states can be divided into blocks with their corresponding keys and regard the hidden state blocks as memory locations [17]. One way to have memory that encodes relationships is to use dot-product attention between memory and input. The memory here is intended to encode relationships [18]. Previously mentioned models have either memory that blends items and relationships together. However, making a separate split of memory to item memory and relational memory with interaction [1] gives state-of-the art performance in a set of toy tasks [19] designed to stress neural networks in the domain of Question and Answering (QA). Among these work, the memory is meant to store something useful for the task. It

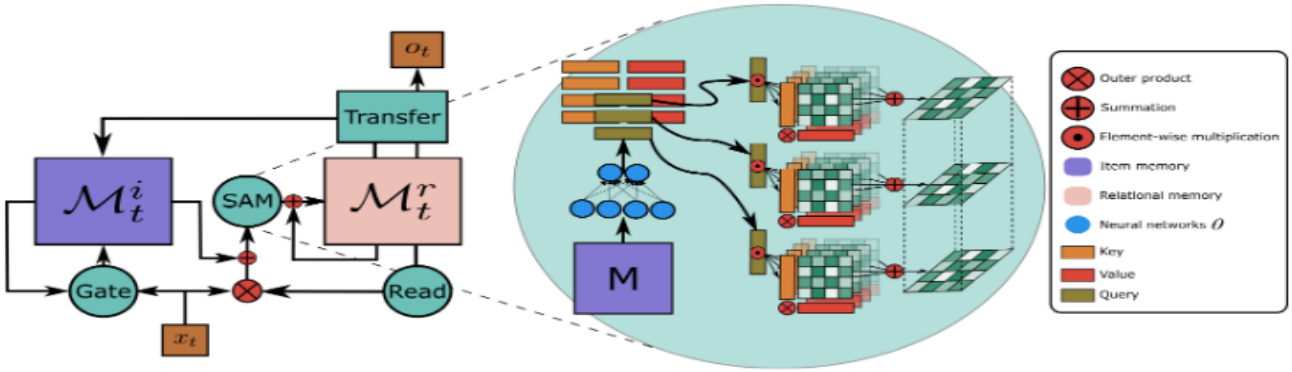


Fig. 1: SAM-Based Two Memory Model. Reprinted from [1].

could be beneficial to guide the model to store particular information. Therefore, we are attempting to force the memory to encode some information obtained from the input. By making a calendar appointment task, the model needs to fill in the right slot according to an incoming appointment. The calendar state is maintained and updated across timesteps. With intended representation, the encoding of the information could be useful for us to understand the memory and establish a new kind of connection from the model to external source of data.

II. Calendar Appointment Task

The task requires the model to output the calendar after being given an input appointment in each timestep. The dataset contains two following components:

1) **Calendar:** A calendar is a matrix $\mathbf{C}_t \in \mathrm{R}^{5x8}$ in which each entry indicates a timeslot totally representing 5 days and 8 hours each day.
2) **Appointment:** An appointment is a vector $\mathbf{a}_t \in \mathrm{R}^3$. The vector represents id of one who makes the request, day and hour of the appointment respectively. The id ranges from 1 to 2, day from 0 to 4 and hour from 0 to 7. Therefore $\mathbf{C}_{t,i,j}$ will receive only 0, 1, 2 as the value, where 0 means that the slot is empty.

On each timestep, the model is given an input:

$$i_t = [\text{flatten}(\mathbf{C}_{t-1}), \mathbf{a}_t],$$

where flatten is a function that maps from a matrix $\mathbf{C}_t$ to a vector with 40 dimensionalities. The token "," denotes vector concatenation, C_{t-1} is the calendar state in the previous timestep and a is the input appointment in the current timestep. The groundtruth of the current calendar state is formalized in as follow:

$$\mathbf{C}_{t,\mathbf{a}_{t,2},\mathbf{a}_{t,3}} = \begin{cases} \mathbf{a}_{t,1}, & \text{if } \mathbf{C}_{t-1,\mathbf{a}_{t,2},\mathbf{a}_{t,3}} = 0 \\ \mathbf{C}_{t-1,\mathbf{a}_{t,2},\mathbf{a}_{t,3}}, & \text{otherwise} \end{cases}$$

The dataset consist of the training set and validation set which contain 30000 examples and 1000 examples. In each training example, we set the length to be 1, 5 and 10 for each experiment. For the validation set, the length varies from 1 to 20 to see the generalizability of the model. To avoid data bias, we initialize the calendar state to be uniformly distributed over the integer range [0,3] for the calendar to be updated. For the appointment data, each component is uniformly distributed over its range as well.

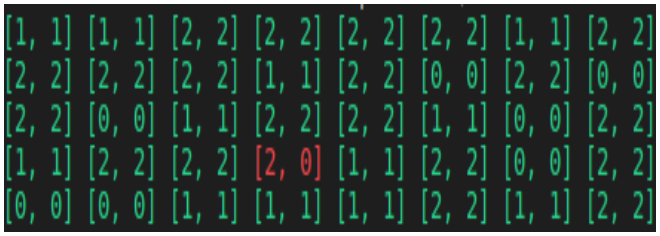
The task is designed to experiment with the neural network with explicit memory to test if it can maintain information in a particular way. Moreover, it also has a condition when a calendar cannot be updated: conflicting timeslot. Therefore, the model that solves the task will be able to prevent faulty data to be stored in the memory: the model has to know that each slot is vacant or occupied, which requires relational reasoning, not just memorization of items. The task could be a way for the model to know how to insert the information to the memory that serves a particular purpose and that might be beneficial as a multi-task learning.

III. Experiment

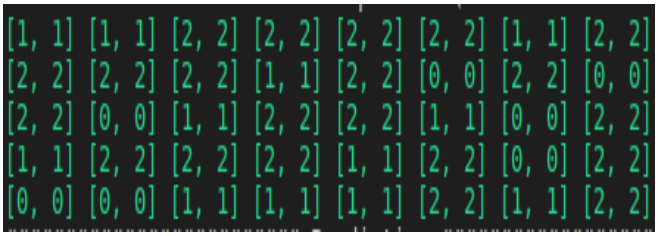
In this section, we perform experiments in different length of dataset. The goal is to see how well the model can pass along the memory state with correct update from appointment input. Here, we explain what the model we use and hyperparameters

A. Model Setup

The model we use for this experiment is STM (SAM-based Two-Memory Model) [1] as shown in Figure 1. The model has two explicit memory: one for items and another for relationship. The input is transformed to two vector by different neural networks. Then, it performs outer-product to get a matrix which will be through the gating mechanism to update the item memory. The relational memory is updated based on its previous read and the content inside the item memory. The relational memory is next extracted information and passed to the item memory. The reason we choose this model is that it is the model that separate the functionalities of memory, making clear where to explicitly store the calendar state. Moreover, the model perform at the state of the art on the bAbI task [19], which shows its ability to memorize data and relate them effectively. Therefore, we can expect the relational memory to check the validity of the appointment



(a) Training calendar state after epoch 1.



(b) Training calendar state after epoch 5.

Fig. 2: Comparison between groundtruth calendar state and the reconstruction of the model. Each pair of values inside a bracket is a groundtruth and reconstruction, respectively. Green color in each slot indicates that the model correctly reconstructs the calendar state. Red color means otherwise.

when being fed to the model. To reconstruct the calendar state, we place a decoder to the item memory. The model has an output module connected to both memory components. However, for our purpose in the experiment, we remove the output module and test the ability of the model to encode the calendar state. We expect that the model should be able to store calendar state in the item memory and update it when there is a valid appointment. We set the task as a classification task at the granularity of each one hot entry in each slot defined as follow:

$$\mathcal{L}_{\theta,t,i,j,k} = y \log \mathbf{C}_{t,i,j,k} + (1-y) \log \mathbf{C}_{t,i,j,k},$$

where y is the groundtruth of the entry k of day i and hour j in the calendar state. Finally, we sum up the loss and take the average as the final loss.

$$\mathcal{L}_{\theta,t} = \frac{\sum_{i,j,k} \mathcal{L}_{\theta,t,i,j,k}}{i * j * k}$$

B. Hyperparameters

We use STM with item memory of the size 150x150, the other hyperparamters are the same as in the bAbI task. The decoder is a 4-Layer MLP. All four layer has 4096, 2048, 1024, and 120 hidden units. The last layer output indicates 5x8 calendar slots each of which contains a one-hot vector of 3 dimensionality indicating emptiness or a person id 1 or 2. We use LeakyReLU activation with 0.1 slope for the negative region of the input. Batch normalization is applied to all layers. The model is trained using Adam optimizer with learning rate 0.0006, beta1 0.9, and beta2 0.99. We use gradient clipping with length 5 which is the default setting for STM in the bAbI task.

IV. Result

As shown in Figure 2a 2b, the model starts from trying to copy the calendar state to the memory with incorrect insert. The appointment input here is [2, 3, 3]. After a number of epochs, it starts to get the insertion correct. The model reaches 100% accuracy on the training set in a few epoch with about 99.% validation accuracy.

We find that without batch normalization, the model struggle to learn and produce only zeros as output.

Length	Train ACC	Valid ACC
1	100.0	99.3
5	99.99	94.6
10	99.85	94.8

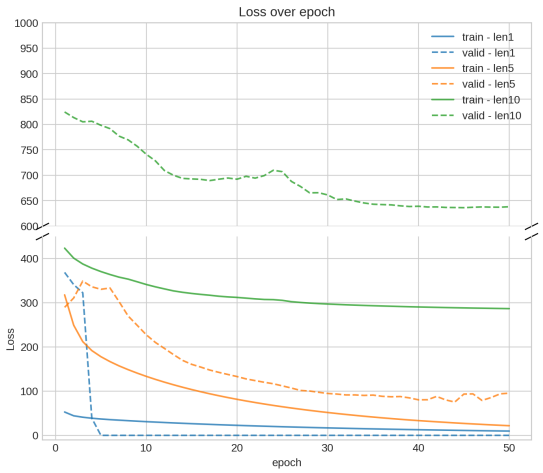
TABLE I: Performance of the model in different lengths.

Table I shows the accuracy result in the length of 1, 5, and 10. The model performs perfectly in length 1 dataset with 99.3% validation accuracy. The model performs almost equally well for the input of the length 5 and 10.

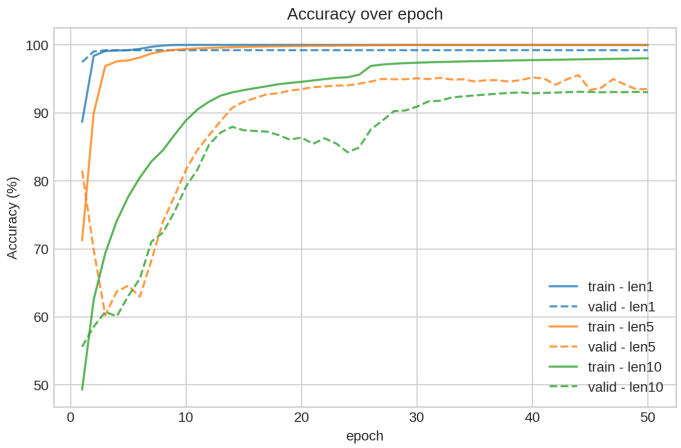
V. Related Work

Synthetic Dataset: Copy and Repeat Copy tasks [12] gives a benchmark to test if a model can correctly copy binary vectors and repeat it for a number of times, which is used to test the ability to memorize item and recall correctly. Associative Recall [12] measures how accurate the model reads the needed information given an input vector sequence and the index query. N^{th} farthest task tests if a model can actually sort the distance between input vectors and the query vector, testing the ability for the model to relate the items inside the memory or the relational memory itself [18]. Not only algorithmic tasks are generated, bAbi task [19], a set of tasks in the domain of Question&Answering (QA), each of which is designed to require a particular set of skill to be able to answer correctly. Relational Associative Recall requires the model to perform a task similar to N^{th} furthest but instead output the vector itself, which tests the ability to keep items and relationships in the memory [1]. All of these tasks are different from ours in that, we attempt to make sure the memory encode the information in the exact we want in every timestep. Explicit memory models might encode something and it can be reconstructed at the end similarly to the Copy Task. However, the reconstructed information is done in the final timestep. Moreover, the encoded information in a particular way may help the model to perform better in another task that require the information, resulting in a multi-task learning.

Memory-Augmented Neural Networks (MANNs): There are mainly two ways researcher improve the memory of neural



(a) The plot shows a loss over epoch.



(b) The plot shows an accuracy over epoch.

Fig. 3: The figure comparing the loss and accuracy over time of the models. a) The loss shows that all models are saturated after 50 epochs. b) At sequence length one, the model can be trained to 100% accuracy with 99.3% on the validation set. At sequence length of 10, it achieved over 98% and 90% on training set and validation set respectively.

networks. The first one is to carefully arrange the hidden states and the architecture. End-to-End Memory Networks [14] transforms the input to embedding vectors as memory locations used to compared to the query to produce an answer. [16] uses sets of GRUs to convert input, query to hidden state representation. Similarity scores are used to compare those two and an additional set of GRUs process the similarity scores ultimately producing episodic memory. EntNet [17] arranges GRU cells as a matrix. Each column has a set of hidden state blocks operating as memory locations with their corresponding keys. Secondly, a separate matrix is used as memory interacting with neural networks through read and write operations. Neural networks can be extended by external memory given as a general framework with the concept of storage and retrieval [13]. Blurry read and write formulate read head and write head mimicking the concept of a computer in an end-to-end differentiable manner [12], [15]. Self dot-product attention can be used to form a memory intended to represent relationships [18]. Separation of memory makes clear the functionality of each module, item memory and relational memory, with different attention mechanisms [1].

Input Reconstruction: Reconstructing input has the idea from two-way conversion from the input to the latent variable and vice versa for the reconstruction of the input image [20]. Getting back input query based on context also improves the performance in QA tasks [21]. Using reconstruction loss with the main task loss shows a better performance in the pathfinding task [22]. In our work, we employ a similar concept but to the explicit memory, which maintain a comprehensive representation of the state while processing the input.

VI. Conclusion and Future Work

We created a synthetic task which is similar to the Copy Task but aditionally requiring the model to update the state in place. Moreover, the model needs to maintain the state across timestep with the purpose of reconstruction. The task is perfected in one timestep with validation accuracy nearly 100% but still struggles with longer timesteps in the validation set. In the future, we plan to make the model able to put any arbitrarily new id to the calendar. Also, making the relational memory decodable in order to put new constraints to the model and let it apply to the information inside the item memory.

Acknowledgment

The authors would like to thank Olivier Nicole for being supportive and making all the machines work smoothly.

References

[1] H. Le, T. Tran, and S. Venkatesh, "Self-attentive associative memory," 08 2020.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[4] K. Cho, B. van Merriënboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 06 2014.

[5] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ArXiv*, vol. 1409, 09 2014.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 06 2017.

[7] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, vol. 4, 09 2014.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[10] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[11] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
[12] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," 10 2014.
[13] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 10 2014.
[14] S. Sukhbaatar, A. D. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *NIPS*, 2015.
[15] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. Gómez, E. Grefenstette, T. Ramalho, J. Agapiou, A. Badia, K. Hermann, Y. Zwols, G. Ostrovski, A. Cain, H. King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, 10 2016.
[16] A. Kumar, O. Irsoy, J. Su, J. Bradbury, R. English, B. Pierce, P. Ondruska, I. Gulrajani, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," 06 2015.
[17] M. Henaff, J. Weston, A. Szlam, A. Bordes, and Y. Lecun, "Tracking the world state with recurrent entity networks," 12 2016.
[18] A. Santoro, R. Faulkner, D. Raposo, J. Rae, M. Chrzanowski, T. Weber, D. Wierstra, O. Vinyals, R. Pascanu, and T. Lillicrap, "Relational recurrent neural networks," 06 2018.
[19] J. Weston, A. Bordes, S. Chopra, A. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," 02 2015.
[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.
[21] M. Seo, S. Min, A. Farhadi, and H. Hajishirzi, "Query-reduction networks for question answering," *arXiv: Computation and Language*, 2017.
[22] B. Peng, Z. Lu, H. Li, and K.-F. Wong, "Towards neural network-based reasoning," *arXiv preprint arXiv:1508.05508*, 2015.