Improving End-to-End Memory Networks with Unified Weight Tying

Fei Liu **Trevor Cohn** **Timothy Baldwin**
School of Computing and Information Systems
The University of Melbourne
Victoria, Australia
fliu3@student.unimelb.edu.au
t.cohn@unimelb.edu.au tb@ldwin.net

Abstract

Answering questions while reasoning over multiple supporting facts has long been a goal of artificial intelligence. Recently, remarkable advances have been made, focusing on reasoning over natural language-based stories. In particular, end-to-end memory networks (N2N), have achieved state-of-the-art results over such tasks. However, N2Ns are limited by the necessity to choose between two weight tying schemes, neither of which performs consistently well over all tasks. We propose a unified model generalising weight tying and in doing so, make the model more expressive. The proposed model achieves uniformly high performance, improving on the best results for memory network-based models on the bAbI dataset, and competitive results on Dialog bAbI.

1 Introduction

Deep neural network models have demonstrated strong performance on a number of challenging tasks, such as image classification (He et al., 2016), speech recognition (Graves et al., 2013), and various natural language processing tasks (Bahdanau et al., 2014; Kim, 2014; Xiong et al., 2016). Recently, the augmentation of neural networks with external memory components has been shown to be a powerful means of capturing context of different types (Graves et al., 2014, 2016; Rae et al., 2016). Of particular interest to this work is the work by Sukhbaatar et al. (2015), on end-to-end memory networks (N2Ns), which exhibit remarkable reasoning capabilities, e.g. for reasoning (Weston et al., 2016) and goal-oriented dialogue tasks (Bordes and Weston, 2016). Typically, such tasks consist of three key components: a sequence of supporting facts (the story), a question, and its answer. An example task is given in Figure 1. Given the first two as input, it is the model's job to reason over the supporting facts and predict the answer to the question.

One drawback of N2Ns is the problem of choosing between two types of weight tying (adjacent and layer-wise; see Section 2 for a technical description). While N2Ns generally work well with either weight tying approach, as reported in Sukhbaatar et al. (2015), the performance is uneven on some difficult tasks. That is, for some tasks, one weight tying approach attains near-perfect accuracy and the other performs poorly, but for other tasks, this trend is reversed.

In this paper, focusing on improving N2N, we propose a unified model, UN2N, capable of dynamically determining the appropriate type of weight tying for a given task. This is realised through the use of a gating vector, inspired by Liu and Perez (2017). Our method achieves the best performance for a memory network-based model on the bAbI dataset, superior to both adjacent and layer-wise weight tying, and competitive results on Dialog bAbI.

The paper is organised as follows: after we review N2N and related reasoning models in Section 2, we describe our motivation and detail the elements of our proposed model in Section 3. Section 4 and 5 present the experimental results on the bAbI and Dialog bAbI datasets with analyses in Section 6. Lastly, Section 7 concludes the paper.

2 Related Work

End-to-End Memory Networks: Building on top of memory networks (Weston et al., 2015), Sukhbaatar et al. (2015) introduced N2N, discard-

Jeff went to the kitchen. Mary travelled to the hallway. Jeff picked up the milk.
Jeff travelled to the bedroom. Jeff left the milk. Jeff went to the bathroom.
Where is the milk now? A: bedroom
Where is Jeff? A: bathroom
Where was Jeff before the bedroom? A: kitchen

Figure 1: Example story, question and answer.

ing the memory position supervision and making the model trainable in an end-to-end fashion, through the advent of supporting memories and a memory access controller. Representations of the context sentences $x_1, \ldots, x_n$ in the story are encoded using two sets of embedding matrices $\boldsymbol{A}$ and $\boldsymbol{C}$ (both of size $d \times |V|$ where d is the embedding size and $|V|$ the vocabulary size), and stored in the input and output memory cells $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n$ and $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$, each of which is obtained via $\boldsymbol{m}_i = \boldsymbol{A}\Phi(x_i)$ and $\boldsymbol{c}_i = \boldsymbol{C}\Phi(x_i)$, where $\Phi(\cdot)$ is a function that maps the input into a bag of dimension $|V|$. The input question q is encoded with another embedding matrix $\boldsymbol{B} \in \mathbb{R}^{d\times|V|}$ such that $\boldsymbol{u} = \boldsymbol{B}\Phi(q)$. `N2N` utilises the question embedding $\boldsymbol{u}$ and the input memory representations $\boldsymbol{m}_i$ to measure the relevance between the question and each supporting context sentence, resulting in a vector of attention weights:

$$p_i = \text{softmax}(\boldsymbol{u}^\top \boldsymbol{m}_i) \tag{1}$$

where $\text{softmax}(a_i) = \frac{e^{a_i}}{\sum_j e^{a_j}}$. Once the attention weights have been computed, the memory access controller receives the response $\boldsymbol{o}$ in the form of a weighted sum over the output memory representations:

$$\boldsymbol{o} = \sum_i p_i \boldsymbol{c}_i \tag{2}$$

To enhance the model's ability to cope with more challenging tasks requiring multiple supporting facts from the memory, Sukhbaatar et al. (2015) further extended the model by stacking multiple memory layers (also known as "hops"), in which case the output of the k^{th} hop is taken as input to the $(k+1)^{\text{th}}$ hop:

$$\boldsymbol{u}^{k+1} = \boldsymbol{o}^k + \boldsymbol{u}^k \tag{3}$$

Lastly, `N2N` predicts the answer to question q using a softmax function:

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W}(\boldsymbol{o}^K + \boldsymbol{u}^K)) \tag{4}$$

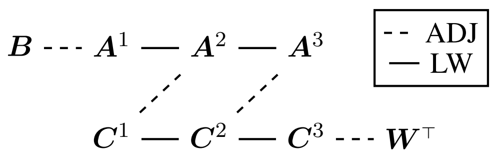

Figure 2: Illustration of the two types of weight tying mechanisms based on a `N2N` model with 3 hops. Lines and dashed lines indicate parameter sharing relationships between embedding matrices.

where $\hat{\boldsymbol{y}}$ is the predicted answer distribution, $\boldsymbol{W} \in \mathbb{R}^{|V|\times d}$ is a parameter matrix for the model to learn (note that in the context of `bAbI` tasks, answers are single words), and K is the total number of hops.

Current issues and motivation: In Sukhbaatar et al. (2015), two types of weight tying were explored for `N2N`, namely adjacent ("`ADJ`") and layer-wise ("`LW`"). With `LW`, the input and output embedding matrices are shared across different hops (i.e., $\boldsymbol{A}^1 = \boldsymbol{A}^2 = \ldots = \boldsymbol{A}^K$ and $\boldsymbol{C}^1 = \boldsymbol{C}^2 = \ldots = \boldsymbol{C}^K$), resembling RNNs. With `ADJ`, on the other hand, not only is the output embedding for a given layer shared with the corresponding input embedding (i.e., $\boldsymbol{A}^{k+1} = \boldsymbol{C}^k$), the answer prediction matrix $\boldsymbol{W}$ and question embedding matrix $\boldsymbol{B}$ are also constrained such that $\boldsymbol{W}^\top = \boldsymbol{C}^K$ and $\boldsymbol{B} = \boldsymbol{A}^1$.

While both `ADJ` and `LW` work well, achieving comparable overall performance in terms of mean error over the 20 `bAbI` tasks, their performance on a subset of the tasks (i.e., tasks 3, 16, 17 and 19, as shown in Table 1) is inconsistent, with one performing very well, and the other performing poorly. Based on this observation, we propose a unified weight tying mechanism exploiting the benefits of both `ADJ` and `LW`, and capable of dynamically determining the best weight tying approach for a given task.

Task	N2N	
	ADJ	LW
3: 3 supporting facts	90.7	**97.9**
16: basic induction	**99.6**	48.2
17: positional reasoning	59.3	**81.4**
19: path finding	33.5	**97.7**

Table 1: Accuracy (%) reported in (Sukhbaatar et al., 2015) on a selected subset of the 20 bAbI 10k tasks. Note that performance in the LW column is obtained with a larger embedding size $d = 100$ and ReLU non-linearity applied to the internal state after each hop.

Related reasoning models: Gated End-to-End Memory Networks (GN2Ns) (Liu and Perez, 2017) are a variant of N2N with a simple yet effective gating mechanism on the connections between hops, allowing the model to dynamically regulate the information flow between the controller and the memory. Dynamic Memory Networks (DMNs) and its improved version (DMN+) employ RNNs to sequentially process contextual information stored in the memory. All these models have been shown to have competent reasoning capabilities over the bAbI dataset (Weston et al., 2016).

3 Proposed Model[1]

The key idea in this work is to let the model determine which type of weight tying mechanism it should rely on. Recall that there are two types: ADJ and LW. With ADJ, the input embedding ($\boldsymbol{A}^k$) of the k^{th} hop is constrained to share the same parameters with the output embedding ($\boldsymbol{C}^{k-1}$) of the $(k-1)^{\text{th}}$ hop (i.e., $\boldsymbol{A}^k = \boldsymbol{C}^{k-1}$). In contrast, with LW, the same input/output embedding matrices are shared across different hops (i.e., $\boldsymbol{A}^1 = \boldsymbol{A}^2 = \ldots = \boldsymbol{A}^K$ and $\boldsymbol{C}^1 = \boldsymbol{C}^2 = \ldots = \boldsymbol{C}^K$). To this end, we design a dynamic mechanism, allowing the model to decide on the preferred type of weight tying based on the input. Specifically, the key element in UN2N is that embedding matrices are constructed dynamically for each instance. This is in contrast to N2N and GN2N where the same embedding matrices are used for every input. UN2N, utilising a gating vector $\boldsymbol{z}$ (described in Equation (8)), constructs the embedding matrices (i.e., $\boldsymbol{A}^k, \boldsymbol{C}^k, \boldsymbol{B}$ and $\boldsymbol{W}$) on the fly, influenced by the information carried by $\boldsymbol{z}$ regarding the input question $\boldsymbol{u}^0$ as well as the context sentences in the story $\boldsymbol{m}_t$:

$$\boldsymbol{A}^{k+1} = \boldsymbol{A}^k \odot \boldsymbol{z} + \boldsymbol{C}^k \odot (\boldsymbol{1} - \boldsymbol{z}) \tag{5}$$

$$\boldsymbol{C}^{k+1} = \boldsymbol{C}^k \odot \boldsymbol{z} + \tilde{\boldsymbol{C}}^{k+1} \odot (\boldsymbol{1} - \boldsymbol{z}) \tag{6}$$

where $\odot$ is the column element-wise multiplication operation, and $\tilde{\boldsymbol{C}}^{k+1}$ the unconstrained embedding matrix. We further define $\boldsymbol{A}^1 = \tilde{\boldsymbol{A}}^1$ and $\boldsymbol{C}^1 = \tilde{\boldsymbol{C}}^1$, where $\tilde{\boldsymbol{A}}^1$ and $\tilde{\boldsymbol{C}}^1$ are the unconstrained embedding matrices for hop 1. As shown in Equation (5), the input embedding matrix $\boldsymbol{A}^{k+1}$ for the $(k+1)^{\text{th}}$ hop is composed of a weighted sum of the input and output embedding matrix $\boldsymbol{A}^k$ and $\boldsymbol{C}^k$ for the k^{th} hop, resembling LW and ADJ, respectively. The summation is weighted by the gating vector $\boldsymbol{z}$. $\boldsymbol{C}^{k+1}$ is constructed in a similar fashion. Ultimately, the larger the values of the elements of $\boldsymbol{z}$, the more UN2N leans towards LW. Conversely, smaller $\boldsymbol{z}$ values indicate an inclination for ADJ.

Another key to this approach is the gating vector $\boldsymbol{z}$, which is formulated to incorporate knowledge informative to the choice of the weight tying scheme. Here, we take inspiration from gated end-to-end memory networks (“GN2N”: Liu and Perez (2017)), where a gating mechanism is learned in an end-to-end fashion to regulate the information flow between memory hops. In GN2N, the gate value $\mathrm{T}^k(\boldsymbol{u}^k)$ depends only on the input to the k^{th} hop $\boldsymbol{u}^k$, whereas in this work, we further condition the determination of the weight tying approach on the story (or memory). Concretely, similarly to DMN and DMN+, we encode the story by first reading the memory one step at a time with a GRU:

$$\boldsymbol{h}_{t+1} = \mathrm{GRU}(\boldsymbol{m}_t, \boldsymbol{h}_t) \tag{7}$$

where t is the current time step and $\boldsymbol{m}_t$ is the context sentence in the story at time t. Then, the last hidden state $\boldsymbol{h}_T$ of the GRU is taken to be the representation of the story.[2] Next, $\boldsymbol{z}$ is defined to be a vector of dimension d:

$$\boldsymbol{z} = \sigma\left(\boldsymbol{W}_z \begin{bmatrix} \boldsymbol{u}^0 \\ \boldsymbol{h}_T \end{bmatrix} + \boldsymbol{b}_z\right) \tag{8}$$

where $\boldsymbol{W}_z$ is a weight matrix, $\boldsymbol{b}_z$ a bias term ,σ the sigmoid function and $\begin{bmatrix} \boldsymbol{u}^0 \\ \boldsymbol{h}_T \end{bmatrix}$ the concatenation

[1]For better readability, a summary table of notations used in this paper is in Table 2.

[2]We also performed additional experiments with a bi-directional GRU over the memory as in Xiong et al. (2016), but did not observe any performance gain.

Var.	Description	Dim.
d	Dimensionality of embeddings	$\mathbb{R}$
$\lvert V\rvert$	Vocabulary size	$\mathbb{R}$
$\boldsymbol{m}_i^k$	Input memory embedding of the i^{th} context sentence at the k^{th} hop	$\mathbb{R}^d$
$\boldsymbol{c}_i^k$	Output memory embedding of the i^{th} context sentence at the k^{th} hop	$\mathbb{R}^d$
$\boldsymbol{u}^k$	Question embedding at the k^{th} hop	$\mathbb{R}^d$
$\boldsymbol{o}^k$	Weighted output embedding at the k^{th} hop	$\mathbb{R}^d$
$\boldsymbol{z}$	Gating vector to dynamically determine the type of weight tying	$\mathbb{R}^d$
$\boldsymbol{h}_t$	Hidden GRU representation of processed memory at step t	$\mathbb{R}^d$
$\boldsymbol{b}_z$	Bias term for computing $\boldsymbol{z}$	$\mathbb{R}^d$
$\boldsymbol{A}^k$	Embedding matrix for input memory cell at the k^{th} hop	$\mathbb{R}^{d\times\lvert V\rvert}$
$\boldsymbol{C}^k$	Embedding matrix for output memory cell at the k^{th} hop	$\mathbb{R}^{d\times\lvert V\rvert}$
$\tilde{\boldsymbol{A}}^k$	Unconstrained embedding matrix for input memory cell at the k^{th} hop	$\mathbb{R}^{d\times\lvert V\rvert}$
$\tilde{\boldsymbol{C}}^k$	Unconstrained embedding matrix for output memory cell at the k^{th} hop	$\mathbb{R}^{d\times\lvert V\rvert}$
$\boldsymbol{B}$	Embedding for question at the 1^{st} hop	$\mathbb{R}^{d\times\lvert V\rvert}$
$\boldsymbol{W}$	Weight matrix for the classifier at the last (K^{th}) hop	$\mathbb{R}^{\lvert V\rvert\times d}$
$\boldsymbol{W}_z$	Weight matrix for computing $\boldsymbol{z}$	$\mathbb{R}^{d\times 2d}$
$\boldsymbol{H}$	Weight matrix for linear transformation between hops	$\mathbb{R}^{d\times d}$

Table 2: Notation summary.

of $\boldsymbol{u}^0$ and $\boldsymbol{h}_T$. Essentially, the gating vector $\boldsymbol{z}$ is now dependent on not only the question $\boldsymbol{u}^0$, but also the context sentences in the memory encoded in $\boldsymbol{h}_T$. Note that the gating vector $\boldsymbol{z}$ can be replaced by a gating scalar z, but we choose to use a vector for more fine-grained control as in LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014).

To simplify the model, we constrain $\boldsymbol{B}$ and $\boldsymbol{W}^\top$ to share the same parameters as $\boldsymbol{A}^1$ and $\boldsymbol{C}^K$. Moreover, following Sukhbaatar et al. (2015), we add a linear mapping $\boldsymbol{H} \in \mathbb{R}^{d\times d}$ to the update connection between memory hops, but in our case, down-weight it by $\boldsymbol{1} - \boldsymbol{z}$, resulting in:

$$\boldsymbol{u}^{k+1} = \boldsymbol{o}^k + (\boldsymbol{H} \odot (\boldsymbol{1} - \boldsymbol{z}))\boldsymbol{u}^k \tag{9}$$

Regularisation: In order to prevent the input and output embedding matrices $\boldsymbol{A}^k$ and $\boldsymbol{C}^k$ from being dominated by the unconstrained embedding matrices, it is necessary to restrain the magnitude of the values in $\tilde{\boldsymbol{A}}^1$ and $\tilde{\boldsymbol{C}}^k$. Therefore, in addition to the cross entropy loss over N training instances:

$$\mathcal{L} = \sum_N \text{CrossEntropy}(\boldsymbol{y}, \hat{\boldsymbol{y}}) \tag{10}$$

where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$ are the true and predicted answer, we enforce a regularisation penalty and formulate the new objective function as:

$$\mathcal{L}' = \mathcal{L} + \lambda \cdot (||\tilde{\boldsymbol{A}}^1||_2^2 + \sum_k ||\tilde{\boldsymbol{C}}^k||_2^2) \tag{11}$$

Model implementation: we implement `UN2N` with TensorFlow (Abadi et al., 2015) and the code is available at `https://github.com/liufly/umemn2n`.

4 QA `bAbI` Experiments

In this section, the experimental setup is detailed, followed by the results.

4.1 Experimental Setup

Dataset: We evaluate the proposed model over the `bAbI` dataset (Weston et al., 2016) (v1.2), featuring 20 different natural-language-based reasoning tasks in the form of: (1) a list of supporting statements ($x_1, \ldots, x_n$); (2) a question (q); and (3) the answer (y, typically a single word or short phrase). Each task in `bAbI` is synthetically generated with a distinct emphasis on a specific type of reasoning. In order to predict the desired answer, the model is required to locate (or focus on) the relevant context sentences among irrelevant distractors in the memory. As with `N2N`, our model can be trained in a fully end-to-end fashion and requires only the answers themselves as the supervision signal. The dataset comes in two sizes, with

Task	N2N		DMN	DMN+	GN2N	UN2N
	ADJ	LW				
1: 1 supporting fact	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
2: 2 supporting facts	99.7	99.7	98.2	99.7	**100.0**	99.2
3: 3 supporting facts	90.7	97.9	95.2	**98.9**	95.5	95.5
4: 2 argument relations	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
5: 3 argument relations	99.4	99.2	99.3	99.5	**99.8**	99.3
6: yes/no questions	**100.0**	99.9	**100.0**	**100.0**	**100.0**	**100.0**
7: counting	96.3	98.0	96.9	97.6	98.2	**98.9**
8: lists/sets	99.2	99.1	96.5	**100.0**	99.7	99.5
9: simple negation	99.2	99.7	**100.0**	**100.0**	**100.0**	**100.0**
10: indefinite knowledge	97.6	**100.0**	97.5	**100.0**	99.8	**100.0**
11: basic coreference	**100.0**	99.9	99.9	**100.0**	**100.0**	**100.0**
12: conjunction	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
13: compound coreference	**100.0**	**100.0**	99.8	**100.0**	**100.0**	**100.0**
14: time reasoning	**100.0**	99.9	**100.0**	**100.0**	**100.0**	**100.0**
15: basic deduction	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
16: basic induction	99.6	48.2	99.4	54.7	**100.0**	99.9
17: positional reasoning	59.3	81.4	59.6	**95.8**	72.2	90.7
18: size reasoning	93.3	94.7	95.3	97.9	91.5	**99.4**
19: path finding	33.5	97.7	34.5	**100.0**	69.0	84.0
20: agent's motivation	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Average	93.4	95.8	93.6	97.2	96.3	**98.3**

Table 3: Accuracy (%) on the 20 bAbI 10k tasks for our proposed method (UN2N) and various benchmark methods. **Bold** indicates the best result for a given task.

either 1k or 10k training instances per task. In this work, we focus exclusively on the 10k version.[3]

Training Details: Following Sukhbaatar et al. (2015), we hold out 10% of the bAbI training set to form a development set. Position encoding and temporal encoding (with 10% random noise) are also incorporated into the model. Training is performed over 100 epochs with a batch size of 32 using the Adam optimiser (Kingma and Ba, 2015) with a learning rate of 0.005. Following Sukhbaatar et al. (2015), linear start is employed in all our experiments for the first 20 epochs. All weight parameters are initialised based on a Gaussian distribution with zero mean and $\sigma = 0.1$. Gradients with an ℓ_2 norm of 40 are divided by a scalar to have norm 40. Also following Sukhbaatar et al. (2015), we use only the most recent 50 sentences as the memory and set the number of memory hops to 3, the embedding size to 20, and λ to 0.001.

Consistent with other published results over bAbI (Sukhbaatar et al., 2015; Graves et al., 2016; Seo et al., 2017), we repeat training 30 times for each task, and select the model which performs best on the development set.

4.2 Results

The results on the 20 bAbI QA tasks are presented in Table 3. We benchmark against other memory network based models: (1) N2N with ADJ and LW (Sukhbaatar et al., 2015); (2) DMN (Kumar et al., 2016) and its improved version DMN+ (Xiong et al., 2016); and (3) GN2N (Liu and Perez, 2017).

Major improvements on the difficult tasks. The most noticeable performance gains are over tasks 16, 17 and 19 where, compared with the vanilla N2N, UN2N achieves much better results than the worst of ADJ and LW, surpassing both ADJ and LW in the case of tasks 17 and 18. This confirms the validity of the model.

[3]We also conducted experiments on the 1k dataset but observed weak performance. We suspect that this is largely due to overfitting given the added complexity of UN2N compared with N2N.

UN2N maintains equally competitive performance on the other tasks. Our unified weight tying scheme does not degrade performance on the less challenging tasks.

Best performing memory network on bAbI 10k. UN2N achieves the best combined results over bAbI 10k, superior to the previous top model DMN+ where two GRUs are employed to process the memory at sentence and hop level; there is a particularly big improvement on task 16 (UN2N = 99.9 vs. DMN+ = 54.7). It is worth noting that UN2N achieves this with a much smaller embedding size $d = 20$ compared to 80 in DMN+.

Comparison with the state-of-the-art. Seo et al. (2017) report a higher overall score (99.3) for a deep learning model which is unrelated to memory networks and thus not immediately comparable with this work. It should also be noted that their model is based on embeddings of size $d = 200$, much larger than UN2N's 20, and that when their model is restricted to an embedding size of $d = 50$, the reported result is 96.8, lower than ours with $d = 20$.

5 Dialog bAbI Experiments

In addition to the experiments on the natural language-based QA bAbI dataset in Section 4, we conduct further experiments on a goal-oriented, dialog-based dataset: Dialog bAbI (Bordes and Weston, 2016).

5.1 Experimental Setup

Dataset: In this work, we employ a collection of goal-oriented dialog tasks, Dialog bAbI, all in a restaurant reservation scenario, developed by Bordes and Weston (2016), consisting of 6 categories each with a specific focus on tasking on aspect of an end-to-end dialog system: 1. issuing API calls, 2. updating API calls, 3. displaying options, 4. providing extra-information, 5. conducting full dialogs (the aggregation of the first 4 tasks), 6. Dialog State Tracking Challenge 2 corpus (DSTC-2). Task 1-5 are generated synthetically in the form of conversation between a user and a bot with entities drawn from a knowledge base with facts defining restaurants and their associated properties (e.g., location and price range, 7 properties in total). Starting with a request from the user, a dialog proceeds with subsequent and alternating user-bot utterances. The bot (or system) needs to figure out the user intention and answer (or react) accordingly. A separate collection of test sets, with entities not occurring in the training set, have also been developed to evaluate the ability of the bot to deal with out-of-vocabulary (OOV) items. Task 6 is based on and derived from the second Dialog State Tracking Challenge (Henderson et al., 2014) with real human-bot conversations.

Training Details: Following the works of (Bordes and Weston, 2016) and (Liu and Perez, 2017), we frame the task in the same fashion: at the t-th time step, the preceding sequence of utterances, $c_1^u, c_1^r, c_2^u, c_2^r, \dots, c_{t-1}^u, c_{t-1}^r$ (alternating between the user request, denoted c_i^u and the system response, denoted c_i^r), is stored in the memory as $\boldsymbol{m}_i$ and $\boldsymbol{c}_i$. Taking the memory as contextual evidence, the goal of the model is to offers an answer c_t^r (the bot utterance at time t) to the question c_t^u (the user utterance at time t).

It is important to notice that the answers in this dataset may no longer be a single but can be comprised of multiple ones. Following (Bordes and Weston, 2016), we replace the final prediction step in Equation (4) with:

$$\hat{\boldsymbol{a}} = \text{softmax}(\boldsymbol{u}^\top \boldsymbol{W'} \Phi(\boldsymbol{y}_1), \dots, \boldsymbol{u}^\top \boldsymbol{W'} \Phi(\boldsymbol{y}_{|C|}))$$

where $\boldsymbol{W'} \in \mathbb{R}^{d \times |V|}$ is the weight parameter matrix for the model to learn, $\boldsymbol{u} = \boldsymbol{o}^K + \boldsymbol{u}^K$ (K is the total number of hops), $\boldsymbol{y}_i$ is the i^{th} response in the candidate set C such that $\boldsymbol{y}_i \in C$, $|C|$ the size of the candidate set, and $\Phi(\cdot)$ a function which maps the input text into a bag of dimension $|V|$.

Additionally, we also append several key features to Φ, following (Bordes and Weston, 2016) and (Liu and Perez, 2017). First, we mark the identity of the speaker of a given utterance (either user or bot). Second, we extend Φ by 7 additional features, one for each of the 7 properties associated with a restaurant. Each of these 7 features indicates whether there are any exact matches between words in the candidate and those in the question or memory. We refer to these 7 features as the *match* features.

In terms of the training procedure, experiments are carried out with the same configuration as described in Section 4.1. As a large variance can be observed due to how sensitive memory-based models are to parameter initialisation, following (Sukhbaatar et al., 2015) and (Liu and Perez, 2017), we repeat each training 10 times using the

Task	–match			+match		
	N2N	GN2N	UN2N	N2N	GN2N	UN2N
1. Issuing API calls	99.9	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
2. Updating API calls	**100.0**	**100.0**	**100.0**	98.3	**100.0**	**100.0**
3. Displaying options	**74.9**	**74.9**	**74.9**	**74.9**	**74.9**	**74.9**
4. Providing information	**59.5**	57.2	57.2	**100.0**	**100.0**	**100.0**
5. Full dialogs	96.1	96.3	**99.2**	93.4	98.0	**99.4**
Average	86.1	85.7	**86.3**	93.3	94.6	**94.9**
1. (OOV) Issuing API calls	72.3	82.4	**83.0**	96.5	**100.0**	**100.0**
2. (OOV) Updating API calls	**78.9**	**78.9**	**78.9**	**94.5**	94.2	94.4
3. (OOV) Displaying options	74.4	**75.3**	75.2	75.2	75.1	**75.3**
4. (OOV) Providing information	**57.6**	57.0	57.0	**100.0**	**100.0**	**100.0**
5. (OOV) Full dialogs	65.5	66.7	**67.8**	77.7	79.4	**79.5**
Average	69.7	72.1	**72.4**	88.8	89.7	**89.8**
6. Dialog state tracking 2	41.1	**47.4**	42.4	41.0	**48.7**	42.9

Table 4: Per-response accuracy on the `Dialog bAbI` tasks. `N2N`: (Bordes and Weston, 2016). `GN2N`: (Liu and Perez, 2017). +match suggests the use of the *match* features in Section 5.1. **Bold** indicates the best result in each group (with or without the *match* features) for a given task.

same hyper-parameters and choose the best system based on validation performance.

5.2 Results

The results on the `Dialog bAbI` tasks are shown in Table 4. In terms of baselines, we benchmark against other memory network-based models:[4] (1) `N2N` (Sukhbaatar et al., 2015); and (2) `GN2N` (Sukhbaatar et al., 2015). While the results of `GN2N` is achieved with `ADJ`, the type of weight tying for `N2N` is not reported in (Bordes and Weston, 2016).

Improvements on task 5. It can be observed that `UN2N` offers consistent performance boost on task 5 across all experiments settings, especially in the non-OOV group. Given that task 5 is the aggregation of the first 4 tasks, the performance increase suggests that the hybrid weight-tying mechanism in `UN2N` is better capable of coping with tasks of various nature.

Equally competitive performance on task 1-4. `UN2N` achieves comparable, if not slightly better in some cases, performance on task 1-4.

[4] Seo et al. (2017) report a higher accuracy on `Dialog bAbI`. However, their model, based on RNNs, is rather different from memory networks and therefore deemed not immediately comparable and unrelated to the goal of this work: improving memory network-based models.

Performance on task 6. Compared to `N2N`, `UN2N` improves the performance consistently with or without the match features. In contrast to `GN2N`, however, this is not the case. The cause for this performance gap requires further investigation and we leave this exercise for future work.

6 Analysis

To gain a better understanding of what the model has learned, we visualise the gating vectors z trained on the difficult `bAbI` tasks (i.e., 3, 16, 17 and 19), in the form of a 2-d PCA scatter plot in Figure 3. Four distinct clusters, representing the 4 different tasks, are easily identifiable. Moreover, it can be observed that tasks 3, 17 and 19 are rather close, reflecting the fact that `LW` performs better than `ADJ` over these three tasks. That is, Figure 3 is further evidence that our model dynamically learns the best weight tying method for a given task.

7 Conclusion

In this paper, we have presented `UN2N`, a model based on `N2N` with a unified weight tying scheme and demonstrated the effectiveness of the proposed method on a set of natural-language-based reasoning and dialog tasks.

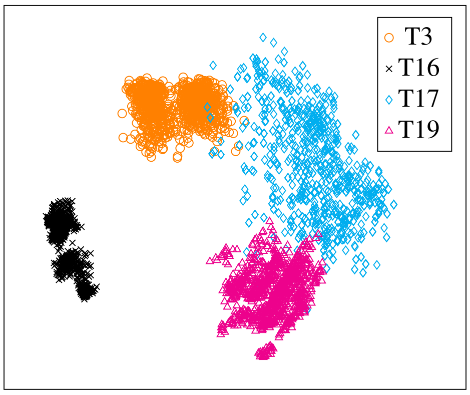

Figure 3: PCA scatter plot of the gating vectors z over tasks 3, 16, 17 and 19.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback, and gratefully acknowledge the support of Australian Government Research Training Program Scholarship. This work was also supported in part by the Australian Research Council.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. https://www.tensorflow.org/.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*. San Diego, USA.

Antoine Bordes and Jason Weston. 2016. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683* .

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar, pages 103–111.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013)*. Vancouver, Canada, pages 6645–6649.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing machines. *arXiv preprint arXiv:1410.5401* .

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538(7626):471–476.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*. Las Vegas, USA.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL 2014)*. Philadelphia, USA, pages 263–272.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. Doha, Qatar.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3th International Conference on Learning Representations (ICLR 2015)*. San Diego, USA.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*. New York, USA.

Fei Liu and Julien Perez. 2017. Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*. Valencia, Spain, pages 1–10.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Tim Lillicrap. 2016. Scaling memory-augmented neural networks with sparse reads and writes. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2016)*. Barcelona, Spain, pages 3621–3629.

Minjoon Seo, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Query-reduction networks for question answering. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS 2015)*. Montréal, Canada, pages 2440–2448.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*. San Juan, Puerto Rico.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*. San Diego, USA.

Caiming Xiong, Stephen Merity, and Richard Socher. 2016. Dynamic memory networks for visual and textual question answering. In *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*. New York, USA, pages 2397–2406.