Multi-Level Memory for Task Oriented Dialogs

Revanth Reddy[1]*, **Danish Contractor**[2], **Dinesh Raghu**[2], **Sachindra Joshi**[2]
[1]Indian Institute of Technology, Madras [2]IBM Research AI, New Delhi
g.revanthreddy111@gmail.com
{dcontrac,diraghu1,jsachind}@in.ibm.com

Abstract

Recent end-to-end task oriented dialog systems use memory architectures to incorporate external knowledge in their dialogs. Current work makes simplifying assumptions about the structure of the knowledge base (such as the use of triples to represent knowledge) and combines dialog utterances (context), as well as, knowledge base (KB) results, as part of the same memory. This causes an explosion in the memory size, and makes reasoning over memory, harder. In addition, such a memory design forces hierarchical properties of the data to be fit into a triple structure of memory. This requires the memory reader to learn how to infer relationships across otherwise connected attributes.

In this paper we relax the strong assumptions made by existing architectures and use separate memories for modeling dialog context and KB results. Instead of using triples to store KB results, we introduce a novel multi-level memory architecture consisting of cells for each query and their corresponding results. The multi-level memory first addresses queries, followed by results and finally each key-value pair within a result. We conduct detailed experiments on three publicly available task oriented dialog data sets and we find that our method conclusively outperforms current state-of-the-art models. We report a 15-25% increase in both entity F1 and BLEU scores.

1 Introduction

Task oriented dialog systems are designed to complete a user specified goal, or service an information request using natural language exchanges. Unlike open domain end-to-end neural dialog models, task oriented systems rely on external knowledge sources, outside of the current conversation context, to return a response (Henderson et al., 2014a; Su et al., 2016; Bordes and Weston, 2017a; Eric and Manning, 2017; El Asri et al., 2017). For instance, in the example shown in Table 1, a dialog agent giving tour package recommendations needs to be able to first query an external knowledge source to determine packages that meet a user's requirement, and then respond accordingly.

Origin	Dest.	Hotel	Price	Cat.	Rating	...
Dallas	Mannheim	Regal Resort	$2800	5.0	8.98	...
Toronto	Calgary	Amusement	$1864.4	4.0	6.91	...
Dallas	Mannheim	Vertex Inn	$3592.8	3.0	7.15	...
Dallas	Santos	Globetrotter	$2000	3.0	8.37	..
Dallas	Mannheim	Starlight	$4018.1	4.0	8.56	
...	...	...	...	...	...	...

Role	Turn	Utterance
Agent	1	how can i help
User	2	i've got a few days off from aug 26-aug 31. I'm not flexible on this, but i want to somehow treat myself with an 8 day trip. I'm leaving Dallas and I want to check out Munich
Agent	2	sorry we dont have any package for Munich
User	3	how about Mannheiem?
Agent	3	would a 5 day trip suffice
User	4	sure dude
Agent	4	A 5 star hotel called the Regal resort, it has free wifi and a spa
User	5	dates?
Agent	5	starts on august 27th until the 30th
User	6	ok that could work. I would like to see my options in Santos.
Agent	6	there is Hotel Globetrotter in Santos. it has 3 stars and comes with breakfast and wifi, it leaves on august 25th and returns on 30th, all for $2000
User	7	ahh i can't leave until august 26th though. I guess i'll just have to go to Mannheim then.
Agent	7	the Regal resort package costs $2800 with economy class.
User	8	yeah i will book it

Table 1: A goal oriented dialog based on the Frames dataset (El Asri et al., 2017) along with an external knowledge source with each row containing a tour package.

In order to enable end-to-end goal oriented dialog tasks, current state of the art methods use neural memory architectures to incorporate external knowledge (Su et al., 2016; Eric and Manning, 2017; Madotto et al., 2018). As can be seen in Table 1, agent responses may also include entity values present only in the dialog context (eg: "Munich" in the Agent response in Turn 2). In order to support such utterances, models also include tokens from the input dialog context in the same memory (Madotto et al., 2018).

*Work done during internship at IBM Research AI

Proceedings of NAACL-HLT 2019, pages 3744–3754
Minneapolis, Minnesota, June 2 - June 7, 2019.

Existing memory based architectures for task oriented dialog suffer from multiple limitations. First, the creation of a shared memory for copying values from dialog context, as well as the knowledge base (KB) results, forces the use of a common memory reader for two different types of data. This makes the task of reasoning over memory, harder – not only does the memory reader need to determine the right entries from a large memory (since each word from context also occupies a memory cell), it also needs to learn to distinguish between the two forms of data (context words and KB results) stored in the same memory.

Subject	Relation	Object	Subject	Relation	Object
Vertex Inn	Price	$3592.8	Vertex Inn	Category	3.0
Regal Resort	Price	$2800	Regal Resort	Rating	8.98
Regal Resort	Category	5.0	Starlight	Price	$4018.1
Starlight	Rating	8.56	Starlight	Category	4.0
...	...	...	...	...	...

Table 2: Results from Dallas to Mannheim stored in the form of triples.

Second, all current neural memory architectures store results, returned by a knowledge source, in the form of triples (eg. $subject - relation - object$). This modeling choice makes it hard for the memory reader to infer relationships across otherwise connected attributes. For instance, consider the example triple store in Table 2 showing results for a query executed for packages between "Dallas" and "Mannheim". If the user asks the dialog agent to check the price of stay at a 5 star hotel, the memory reader needs to infer that the correct answer is $2800 by learning that the price, category and hotel need to be linked inorder to return an answer (shown in blue).

Lastly, current models treat conversations as a sequential process, involving the use of KB results from only the most recent information request/query. In contrast, in real world dialogs such as the one shown in Table 1, the agent may have to refer to results (*to Mannheim*) from a previously executed query (see Turn 7). Thus, at each turn, the system has to memorize all the information exchanged during the dialog, and infer the package being referred to, by the user. In order to support such dialogs, the memory needs to store results of all queries executed during the course of the dialog. The problem of inferring over such results (which may be from multiple queries) is exacerbated when memory is represented in the form of triples.

In this paper, we present our novel multi-level memory architecture that overcomes the limitations of existing methods: (i) We separate the memory used to store tokens from the input context and the results from the knowledge base. Thus, we learn different memory readers for context words as well for knowledge base entities (ii) Instead of using a $subj - rel - obj$ store, we develop a novel multi-level memory architecture which encodes the natural hierarchy exhibited in knowledge base results by storing queries and their corresponding results and values at each level. We first attend on the queries, followed by the results in each query to identify the result being referred to, by the user. We then attend on the individual entries in the result to determine which value to copy in the response. Figure 1c shows our multi-level memory storing the results from queries executed as part of the dialog in Table 1.
Our paper makes the following contributions:
1. We propose the use of separate memories for copying values from context and KB results. Thus, the model learns separate memory readers for each type of data.
2. Our novel multi-level memory for KB results, models the queries, results and their values in their natural hierarchy. As our experiments show, the separation of memory as well as our multi-level memory architecture, both, contribute to significant performance improvements.
3. We present detailed experiments demonstrating the benefit of our memory architecture along with model ablation studies. Our experiments on three publicly available datasets (**CamRest676** (Su et al., 2016), **InCar assistant** (Eric and Manning, 2017), **Maluuba Frames** (El Asri et al., 2017)) show a substantial improvement of 15-25 % in both entity F1 scores, and BLEU scores as compared to existing state of the art architectures. To the best of our knowledge, we are the first to attempt end-to-end modeling of task oriented dialogs with non-sequential references as well as multiple queries, as seen in the Maluuba Frames dataset. A human evaluation on model outputs also shows our model is preferred by users over existing systems such as KVRet (Eric and Manning, 2017) and Mem2Seq (Madotto et al., 2018).

2 Related work

Recent methods, such as (Vinyals and Le, 2015; Serban et al., 2016, 2017), proposed for end-to-end learning of dialogs were aimed at modeling

open-domain dialogs. While they can be used for learning task oriented dialogs, they are not well suited to interface with a structured KB. To better adapt them to handle task oriented dialogs: 1) (Bordes and Weston, 2017b) proposed a memory network based architecture to better encode KB tuples and perform inferencing over them and 2) (Madotto et al., 2018) incorporated copy mechanism to enable copying of words from the past utterances and words from KB while generating responses. All successful end-to-end task oriented dialog networks (Eric and Manning, 2017; Bordes and Weston, 2017b; Madotto et al., 2018) make assumptions while designing the architecture: 1) KB results are assumed to be a triple store, 2) KB triples and past utterances are forced to be represented in a shared memory to enable copying over them. Both these assumptions make the task of inferencing much harder. Any two fields linked directly in the KB tuple are now linked indirectly by the subject of the triples. Further, placing the KB results and the past utterances in same memory forces the architecture to encode them using a single strategy. In contrast, our work uses two different memories for past utterances and KB results. The decoder is equipped with the ability to copy from both memories, while generating the response. The KB results are represented using a multi-level memory which better reflects the natural hierarchy encoded by sets of queries and their corresponding result sets.

Memory architectures have also been found to be helpful in other tasks such as question answering. Work such as (Xu et al., 2016) defines a hierarchical memory architecture consisting of sentence level memory followed by word memory for a QA task while (Chandar et al., 2016) defines a memory structure that speeds up loading and inferencing over large knowledge bases. Recent work by (Chen et al., 2018) uses a variational memory block along with a hierarchical encoder to improve diversity of open domain dialog responses.

3 Multi-Level Memory Network

In this section, we describe our end-to-end model for task oriented dialogues. Our model[1] (Figure 1a) consists of: (i) a hierarchical encoder which encodes the current input context consisting of the user and agent utterances (ii) a multi-level memory that maintains the queries and knowledge base results seen so far in the course of the dialogue, and (iii) copy augmented sequence decoder that uses separate context and multi-level memory. The queries and their corresponding results are maintained in a multi-level memory. The decoder uses a gating mechanism for memory selection while generating a response.

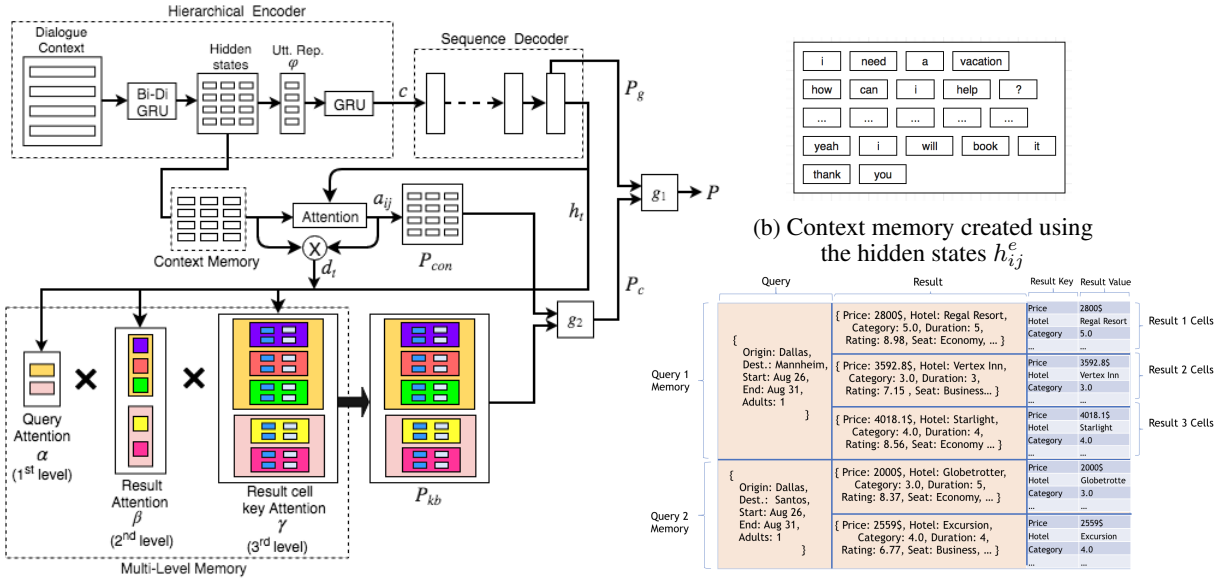
3.1 Encoder

Our model uses a standard hierarchical encoder as proposed by (Sordoni et al., 2015). The encoder takes a sequence of utterances as input. For the t^{th} turn, the dialogue context can be represented as $(c_1, c_2, ...c_{2t-1})$, which consists of t user utterances and $t-1$ system utterances. Each utterance c_i is further a sequence of words $(w_{i1}, w_{i2}, ...w_{im})$. We first embed each word w_{ij} using a word embedding function ϕ^{emb} that maps each word to a fixed-dimensional vector. We then generate utterance representations, $\varphi(c_i)$ using a single layer bi-directional GRU. h^e_{ij} denotes the hidden state of word w_{ij} in the bi-directional GRU. The input representation c is generated by passing each utterance representation $\varphi(c_i)$ through another single layer GRU.

3.2 Multi-level Memory

Motivation: Current approaches break down KB results by flattening them into (*subj-rel-obj*) triples. However, converting KB results into triples leads to loss of relationship amongst attributes in the result set. This makes the reasoning over memory difficult as model now has to infer relationships when retrieving values from memory. Instead, we use a multi-level memory which keeps the natural hierarchy of results intact (without breaking them into triples). We store the queries and their corresponding results and individual values at different *levels*. We first attend on the queries and then on the results for each query to identify which result the user is referring to. This also enables us to handle user requests that refer to results from a previously executed query. We propose that a representation of all the values in the result, and not just one of the values (designated as *subj*), should be used while attending over a result in KB. We attend on this compound representation of the result before attending on the individual key-value pairs in each result, to determine which value to copy into the generated response.

[1] Code is available at Multi-Level Memory



(a) Architecture of our model with multi-level memory attention

(b) Context memory created using the hidden states h^e_{ij}

(c) Expanded view of the multi-level KB memory corresponding to example in Table 1

Figure 1: Model architecture (a) along with schematic representation of context memory (b) and multi-level KB memory (c)

3.2.1 Memory Representation

Let $q_1, q_2, ...q_k$ be the queries fired to the knowledge base till the current state of the dialogue. Every query q_i is a set of key-value pairs $\{k_a^{q_i} : v_a^{q_i}, 1 < a < n_{q_i}\}$, corresponding to the query's slot and argument where n_{q_i} is the number of slots in query q_i. For example, after the user utterance at Turn 3 in Table 1, the query fired by the system on the knowledge base would be *{'origin':'Dallas','destination':'Manheim','Start': 'Aug 26', 'end': 'Aug 31', 'Adults':1}*. The execution of a query on an external knowledge base, returns a set of results. Let r_{ij} be the j^{th} result of query q_i. Each result r_{ij} is also a set of slot-value pairs $\{k_a^{r_{ij}} : v_a^{r_{ij}}, 1 < a < n_{r_{ij}}\}$ where $n_{r_{ij}}$ is the number of attributes in result r_{ij}. A visualization of the memory with queries and their corresponding results can be seen in Figure 1c.

The first level of memory contains the query representations. Each query q_i is represented by q_i^v = Bag of words over the word embeddings of values ($v_a^{q_i}$) in q_i. The second level of memory contains the result representations. Representation of each result r_{ij} is given by r_{ij}^v = Bag of words over the word embeddings of values ($v_a^{r_{ij}}$) in r_{ij}. The third level of memory contains the result cells which have the key-value pairs ($k_a^{r_{ij}} : v_a^{r_{ij}}$) of the results. The values ($v_a^{r_{ij}}$) which are to be copied into the system response are thus present in the final level of memory . We now describe how we apply attention over the context and multi-level memory.

3.3 Decoder

The model generates the agent response word-by-word; a word at time step t is either generated from the decode vocabulary or is a value copied from one of the two memories (knowledge base or context memory). A soft gate g_1 controls whether a value is generated from vocabulary or copied from memory. Another gate g_2 determines which of the two memories is used to copy values.

3.3.1 Generating words:

Let the hidden state of the decoder at time t be h_t.

$$h_t = GRU(\phi^{emb}(y_{t-1}), h_{t-1}) \quad (1)$$

The hidden state h_t is used to apply attention over the input context memory. Attention is applied over the hidden states of the input bi-directional (BiDi) GRU encoder using the "*concat*" scheme as given in (Luong et al., 2015). The attention for the jth word in the ith utterance is given by:

$$a_{ij} = \frac{exp(w_1^T tanh(W_2 tanh(W_3[h_t, h^e_{ij}])))}{\sum_{ij} exp(w_1^T tanh(W_2 tanh(W_3[h_t, h^e_{ij}])))} \quad (2)$$

The attention scores a_{ij} are combined to create an attended context representation d_t,

$$d_t = \sum_{i,j} a_{i,j} h^e_{ij} \quad (3)$$

and similar to (Luong et al., 2015), the decoder word-generation distribution is given by :

$$P_g(y_t) = softmax(W_1[h_t, d_t] + b_1) \quad (4)$$

3.3.2 Copying words from context memory:

The input context memory is represented using the hidden states h^e_{ij} of the input Bi-Di GRU encoder. Similar to (Gulcehre et al., 2016), the attention scores a_{ij}, are used as the probability scores to form the copy distribution $P_{con}(y_t)$ over the input context memory.

$$P_{con}(y_t = w) = \sum_{ij:w_{ij}=w} a_{ij} \quad (5)$$

3.3.3 Copying entries from KB memory:

The context representation d_t, along with the hidden state of decoder h_t, is used to attend over the multi-level memory. The first level attention, $\alpha_.$, is applied over the queries $q_.$.

$$\alpha_i = \frac{exp(w_2^T tanh(W_4[d_t, h_t, q_i^v]))}{\sum_i exp(w_2^T tanh(W_4[d_t, h_t, q_i^v]))} \quad (6)$$

The second level attention, $\beta_{i.}$, is the attention over the results $r_{i.}$ of query q_i.

$$\beta_{ij} = \frac{exp(w_3^T tanh(W_5[d_t, h_t, r_{ij}^v]))}{\sum_j exp(w_3^T tanh(W_5[d_t, h_t, r_{ij}^v]))} \quad (7)$$

The product of first level attention and second level attention is the attention over results of all the queries in the multi-level memory. The weighted sum of the first level attention, second level attention and result representations gives us the attended memory representation, m_t.

$$m_t = \sum_i \sum_j \alpha_i \beta_{ij} r_{ij}^v \quad (8)$$

Each result is further composed of multiple result cells. On the last level of memory, which contains the result cells, we apply key-value attention similar to (Eric and Manning, 2017). The key of the result cell is the word embedding of the slot, $k_a^{r_{ij}}$, in the result. The attention scores, $\gamma_{ij.}$, for the keys represent the attention over the result cells of each result r_{ij}.

$$\gamma_{ijl} = \frac{exp(w_4^T tanh(W_6[d_t, h_t, \phi^{emb}(k_l^{r_{ij}})]))}{\sum_l exp(w_4^T tanh(W_6[d_t, h_t, \phi^{emb}(k_l^{r_{ij}})]))} \quad (9)$$

The product of first level attention α_i, second level attention β_{ij} and third level attention γ_{ijl} gives the final attention score of the value $v_l^{r_{ij}}$ in the KB memory. These final attention scores when combined (Eq. 10), form the copy distribution, $P_{kb}(y_t)$, over the values in KB memory.

$$P_{kb}(y_t = w) = \sum_{ijl:v_l^{r_{ij}}=w} \alpha_i \beta_{ij} \gamma_{ijl} \quad (10)$$

3.3.4 Decoding

Similar to (Gulcehre et al., 2016), we combine the generate and copy distributions - we use gate g_2 (Eq. 11) to obtain the copy distribution $P_c(y_t)$ (Eq. 12) by combining $P_{kb}(y_t)$ and $P_{con}(y_t)$.

$$g_2 = sigmoid(W_7[h_t, d_t, m_t] + b_2) \quad (11)$$

$$P_c(y_t) = g_2 P_{kb}(y_t) + (1 - g_2) P_{con}(y_t) \quad (12)$$

Finally, we use gate g_1 to obtain the final output distribution $P(y_t)$, by combining generate distribution $P_g(y_t)$ and copy distribution $P_c(y_t)$ as shown below:

$$g_1 = sigmoid(W_8[h_t, d_t, m_t] + b_3) \quad (13)$$

$$P(y_t) = g_1 P_g(y_t) + (1 - g_1) P_c(y_t) \quad (14)$$

We train our model by minimizing the cross entropy loss $-\sum_{t=1}^{T} log(P(y_t))$.

4 Experiments

4.1 Datasets

We present our experiments using three real world publicly available multi-turn task oriented dialogue datasets: the InCar assistant (Eric and Manning, 2017), CamRest (Su et al., 2016) and the Maluuba Frames dataset (El Asri et al., 2017). All three datasets contain human-human task oriented dialogues which were collected in a Wizard-of-Oz (Wen et al., 2017) setting.

(i) **InCar assistant dataset** consists of 3031 multi-turn dialogues in three distinct domains: calendar scheduling, weather information retrieval, and point-of-interest navigation. Each dialogue has it's own KB information provided and thus, the system does not have to make any queries.

(ii) **CamRest dataset**, consists of 676 human-to-human dialogues set in the restaurant reservation domain. There are three queryable slots (food, price range, area) that users can specify. This dataset has currently been used for evaluating slot-tracking systems. Recent work by (Lei et al., 2018) uses an end-to-end network without a KB and substitutes slot values with placeholders bearing the slot names in agent responses. However, we formatted the data to evaluate end-to-end systems by adding API call generation from the slot values so that restaurant suggestion task can proceed from the KB results.

(iii) **Maluuba Frames dataset**, consists of 1369 dialogues developed to study the role of memory in task oriented dialogue systems. The dataset is set in the domain of booking travel packages which involves flights and hotels. In contrast to the previous two datasets, this dataset contains dialogs that require the agent to remember all information presented previously as well as support results from multiple queries to the knowledge base. A user's preferences may change as the dialogue proceeds, and can also refer to previously presented queries (non-sequential dialog). Thus, to store multiple queries, we require 3 levels in our multi-level memory as compared to 2 levels in the other datasets, since they don't have more than one query. We do not use the dialogue frame annotations and use only the raw text of the dialogues. We map ground-truth queries to API calls that are also required to be generated by the model. Recent work has used this dataset only for frame tracking (Schulz et al., 2017) and dialogue act prediction (Peng et al., 2017; Tang et al., 2018). To the best of our knowledge we are the first to attempt the end-to-end dialog task using this dataset. Table 3 summarizes the statistics of the datasets.

	InCar	CamRest	Maluuba Frames
Train Dialogs	2425	406	1095
Val Dialogs	302	135	137
Test Dialogs	304	135	137
Avg. no. of turns	2.6	5.1	9.4
Avg length. of sys. resp.	8.6	11.7	14.8
Avg no. of sys. entities	1.6	1.7	2.9
Avg no. of queries	0	1	2.4
Avg no. of KB entries	66.1	13.5	141.2

Table 3: Statistics for 3 different datasets.

4.2 KB API Call Generation

In this section, we briefly describe how the knowledge base queries are generated as API calls as part of the model response. The InCar assistant dataset has a fixed KB for each dialogue whereas the CamRest and Maluuba datasets require queries to be fired on a global KB. Queries in CamRest dataset can have 3 slots namely cuisine, area and pricerange, whereas those in Maluuba can have 8 slots, which are destination, origin, start date, end date, budget, duration, number of adults and children. Any query that is to be fired on the KB is expected to be generated by the model as an API call, by considering a fixed ordering of slots in the generated response. For eg., in CamRest dataset, ApiCall(area=south, pricerange=cheap) would be generated by the model as *api_call dontcare south cheap*, with *dontcare* meaning that the user does not have any preference for cuisine and, *south*, *cheap* being the user constraints for area and pricerange respectively. Therefore, the task of API call generation typically involves copying relevant entities that are present in dialog context.

4.3 Training

Our model is trained end-to-end using Adam optimizer (Kingma and Ba, 2014) with a learning rate of $2.5e^{-4}$. The batch-size is sampled from [8,16]. We use pre-trained Glove vectors (Pennington et al., 2014) with an embedding size of 200. The GRU hidden sizes are sampled from [128, 256]. We tuned the hyper-parameters with grid search over the validation set and selected the model which gives best entity F1.

4.4 Evaluation Metrics

4.4.1 BLEU

We use the commonly used BLEU metric (Papineni et al., 2002) to study the performance of our systems as it has been found to have strong correlation (Sharma et al., 2017) with human judgments in task-oriented dialogs.

4.4.2 Entity F1

To explicitly study the behaviour of different memory architectures, we use the entity $F1$ to measure how effectively values from the knowledge base are used in the dialog. To compute the entity F1, we micro-average the precision and recall over the entire set of system responses to compute the micro F1[2]. For the InCar Assistant dataset, we compute a per-domain entity F1 as well as the aggregated entity F1. Since our model does not have slot-tracking by design, we evaluate

[2]We observe that (Madotto et al., 2018) reports the micro average of recall as the micro F1.

Model	InCar					CamRest		Maluuba Frames	
	BLEU	F1	Calendar F1	Weather F1	Navigate F1	BLEU	F1	BLEU	F1
Attn seq2seq (Luong et al., 2015)	11.3	28.2	36.9	35.7	10.1	7.7	25.3	3.7	16.2
Ptr-UNK (Gulcehre et al., 2016)	5.4	20.4	22.1	24.6	14.6	5.1	40.3	5.6	25.8
KVRet (Eric and Manning, 2017)	13.2	48.0	62.9	47.0	41.3	13.0	36.5	10.7	31.7
Mem2Seq (Madotto et al., 2018)	11.8	40.9	61.6	39.6	21.7	14.0	52.4	7.5	28.5
Multi-level Memory Model (MM)	**17.1**	**55.1**	**68.3**	**53.3**	**44.5**	**15.9**	**61.4**	**12.4**	**39.7**

Table 4: Comparison of our model with baselines

on entity F1 instead of the slot-tracking accuracy as in (Henderson et al., 2014b; Wen et al., 2017)

4.5 Baselines

We experiment with the following baseline models for comparing the performance of our Multi-Level Memory architecture:

Attn seq2seq[3] (Luong et al., 2015): A model with simple attention over the input context at each time step during decoding.

Ptr-UNK[3] (Gulcehre et al., 2016): The model augments a sequence-to-sequence architecture with attention-based copy mechanism over the encoder context.

KVRet (Eric and Manning, 2017): The model uses key value knowledge base in which the KB is represented as triples in the form of $subject - relation - object$. This model does not support copying words from context. The sum of word embeddings of $subject$, $relation$ is used as the key of the corresponding $object$.

Mem2Seq[3] (Madotto et al., 2018): The model uses a memory networks based approach for attending over dialog history and KB triples. During decoding, at each time step, the hidden state of the decoder is used to perform multiple hops over a single memory which contains both dialog history and the KB triples to get the pointer distribution used for generating the response.

4.6 Results

Table 4 shows the performance of our model against our baselines. We find that our multi- level memory architecture comprehensively beats all existing models, thereby establishing new state-of-the- art benchmarks on all three datasets. Our model outperforms each baseline on both BLEU and entity F1 metrics.

InCar: On this dataset, we show entity F1 scores for each of the scheduling, weather and navigation domains. Our model has the highest F1 scores across all the domains. It can be seen that our model strongly outperforms Mem2Seq on each domain. A detailed study reveals that the use of triples cannot handle cases when a user queries with *non-subject* entries or in cases when the response requires inferencing over multiple entries. In contrast, our model is able to handle such cases since we use a compound representation of entire result (bag of words over values) while attending on that result.

CamRest: Our model achieves the highest BLEU and entity F1 scores on this dataset. From Table 4, we see that simpler baselines like Ptr-UNK show competitive performance on this dataset because, as shown in Table 3, CamRest dataset has relatively fewer KB entries. Thus, a simple mechanism for copying from context results in good entity F1 scores.

	InCar		CamRest		Maluuba	
	Ctxt.	KB	Ctxt.	KB	Ctxt.	KB
Mem2Seq	66.2	25.3	63.7	36.5	17.7	8.9
Multi-level Mem.	81.6	37.5	70.1	53.4	27.2	14.6

Table 5: Percentage (%) of category-wise (context vs KB) ground truth entities correctly captured in generated response. Abbreviation Ctxt denotes context.

Maluuba Frames: The Maluuba Frames dataset was introduced for the frame tracking task. Here, a dialog frame is a structured representation of the current dialog state. Instead of explicitly modeling the dialog frames, we use the context representation d_t to directly attend on the Multi-level memory. As Table 3 shows, this dataset contains significantly longer contexts as well as larger number of entities, as compared to the other two datasets. In addition, unlike other datasets, it also contains non-linear dialog flows where a user may refer to previously executed queries and results. The complexity of this dataset is reflected in the relatively lower BLEU and F1 scores as compared to other datasets.

4.7 Analysis

4.7.1 Entity source-wise performance

To further understand the effect of separating context memory from KB memory and using a multi-

[3]We use the implementation provided by (Madotto et al., 2018) at https://github.com/HLTCHKUST/Mem2Seq

	InCar					CamRest		Maluuba Frames	
Model	BLEU	F1	Calendar F1	Weather F1	Navigate F1	BLEU	F1	BLEU	F1
Unified Context and KB memory (Mem2Seq)	11.8	40.9	61.6	39.6	21.7	14.0	52.4	7.5	28.5
Separate Context and KB Memory	14.3	44.2	56.9	54.1	24.0	14.3	55.0	12.1	36.5
+Replace KB Triples with Multi-level memory	17.1	55.1	68.3	53.3	44.5	15.9	61.4	12.4	39.7

Table 6: Model ablation study : Effect of (i) separate memory and (ii) multi-level memory design.

level memory for KB, Table 5 shows the percentage of ground-truth entities, according to their category, which were also present in the generated response. For example, on the InCar dataset, out of the 930 entities in ground-truth response that were to be copied from the KB, our model was able to copy 37.5% of them into the generated response. From Table 5, it can be seen that our model is able to correctly copy a significantly larger number of entities from both, KB and context, as compared to the recent Mem2Seq model in all datasets.

4.7.2 Model ablation study

We report results from ablation studies on all three datasets. Table 6 shows the incremental benefit obtained from individual components used in our model. We investigate the gains made by (i) Using separate memory for context and KB triples (ii) Replacing KB triples with a Multi-level memory. We use the recent Mem2Seq model for comparison with a unified context and KB memory model.

As can be seen from Table 6, the separation of context memory and KB memory leads to a significant improvement in BLEU and F1 scores on all datasets. This validates our hypothesis that storing context words and KB results in a single memory confuses the memory reader. The use of a multi-level memory instead of triples leads to further gains. This suggests, better organization of KB result memory by keeping the natural hierarchy intact is beneficial.

4.7.3 Error Analysis

We analyzed the errors made by our dialog model on 100 dialog samples in test set of Maluuba Frames. We observed that the errors can be divided into five major classes: (i) Model outputs wrong KB result entry due to incorrect attention (27%), (ii) Model returns package details instead of asking for more information from the user (16%), (iii) Model incorrectly captures user intent (13%), (iv) Model makes an error due to non-sequential nature of dialog (22%). In such errors, our model either generates an API call for a result already present in memory, or our model asks for a query-slot value that was already provided by the user, (v) Data specific characteristics such as insufficient samples for certain classes of utterances (eg: more than one package returned) or returning different, but meaningful package attributes as compared to ground-truth data, contribute to 22% of the errors.

	CamRest			**Maluuba**		
	Info.	Lang.	MRR	Info.	Lang.	MRR
KVRet	2.49	4.38	0.57	2.42	3.31	0.64
Mem2Seq	2.48	3.72	0.51	1.78	2.55	0.50
Our Model	**3.62**	**4.48**	**0.76**	**2.45**	**3.93**	**0.69**

Table 7: Informational accuracy and language quality scores using a human study.

4.8 Human Evaluation

We also conducted a blind user study that compared outputs from our model, Mem2Seq and KVRet systems. We used 96 randomly selected examples from each test split of Maluuba and CamRest datasets resulting in a total of 192 examples. Our study was split across 8 users who were provided with results fetched from the KB, current dialog context, gold response and the outputs of each of the models. Model outputs were shuffled in each example and users were asked to score each output between 1 (lowest) to 5 (highest) in terms of its accuracy of information in response and the quality of language. The results of this study are presented in Table 7. We also report the MRR (mean-reciprocal rank) for model preference along with other scores.

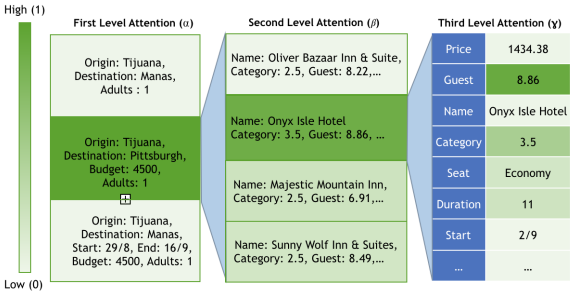
It can be seen that our model consistently ranks high for both information accuracy and language quality as well as reports a higher MRR. To further understand the quality of model performance, we asked the human evaluators whether their best ranked model output was a useful response. We saw that the evaluators agreed in 76.04% and 58.33% of the cases for CamRest and Maluuba datasets respectively. We observe that the results from human evaluation go hand-in-hand with automatic evaluation and reinforce our claim that separating context, KB memory and using a multi-level representation for the KB memory are useful for improving dialog modeling.

Role	Turn	Utterance
Agent	1	hello ! how can i help you today ?
User	2	*i just need to dream a bit. i have 4500 dollars but no vacation days. i'm wondering what a hypothetical trip to Manas would be like*
Agent	2	i can help with that, where are you departing from
User	3	*i'm in Tijuana, not departing any time soon unfortunately though*
...	...	...
Agent	5	`[API_CALL manas tijuana 29/8 16/9 4500 ...]`
...	...	...
User	7	*that's fine, what about pittsburgh*
Agent	7	`[API_CALL pittsburgh tijuana ... 4500 ...]`
...	...	...
Agent	8	that i do have! would you be satisfied with a 3.5 star hotel ?
User	9	*as of now i'm satisfied with a tent and a backpack . what are the hotel details?*
Agent	9	the hotel is situated near a park, comes with free parking, wifi and breakfast ! does this sound good
User	10	*sounds quite nice, what about the guest rating ?*
Gold		the guest rating is impressive : **8.86**
Mem2Seq		this is the only available with the same
KVRet		it's a 2.5 star hotel near a park, museum and airport.it has a guest rating of 8.22
Our Model		the onyx isle hotel has a **8.86** guest rating and offers free parking, breakfast and wifi.

Table 8: Comparing the responses generated by various models on an example in test set of Maluuba Frames.

4.9 Attention Visualization

Analyzing the attention weights is a useful way to understand how the model is inferencing over the memory to copy entities from it. Table 8 shows an example of a dialog from the Maluuba Frames dataset and the outputs generated by different models. Here, the user first wants to know about packages to 'Manas' and then requests for trips to 'Pittsburgh'. Later, the user becomes interested in the 3.5 star hotel in Pittsburgh which was suggested by the agent and wants to know its guest rating. It can be seen from Table 8 that our model outputs the correct guest rating (8.86) of the hotel. Mem2Seq fails to understand the context and generates an irrelevant response. KVRet generates a readable response but points to the guest rating of a different hotel.



(a) Attention over the multi-level KB memory

Word	rating	guest	3.5	1	park	...
Score	0.9132	0.0721	0.0108	0.0030	0.0005	...

(b) Decreasing order of attention scores over words in dialogue context

Figure 2: Visualization of attention over memory while generating the word '8.86' for the example in Table 8.

The attention over the memory while generating the word '8.86' for this example is shown in Fig 2. Fig 2a shows that the query with destination as 'Pittsburgh' gets the highest attention and among the results of this query, the package with the 3.5 star rated hotel gets highest attention. Within this result, the model gives highest score to the result cell with guest rating as the key. To further understand why the correct result hotel gets higher attention, Fig 2b shows the attention scores over the words in context memory. The context representation d_t captures the important words (3.5, guest, rating) in context which are in-turn used to apply attention over the multi-level memory.

Lastly, studying the values of the gates g_1 (prob. of generating from vocabulary) and g_2 (prob. of copying from KB), we found that gate g_1 had a probability value of *0.08* thereby driving the model to copy from memory instead of generating from output vocabulary and gate g_2, with a probability value of *0.99*, was responsible for selecting KB memory over context memory.

5 Conclusion

In this paper, we presented an end-to-end trainable novel architecture with multi-level memory for task oriented dialogues. Our model separates the context and KB memory and combines the attention on them using a gating mechanism. The multi-level KB memory reflects the natural hierarchy present in KB results. This also allows our model to support non-sequential dialogs where a user may refer to a previously suggested result. We find that our model beats existing approaches by 15-25% on both entity F1 and BLEU scores, establishing state-of-the-art results on three publicly available real-world task oriented dialogue datasets. In a user study comparing outputs from our system against recent models, we found that our model consistently scored higher for both language quality as well as correctness of information in the response. We also present the benefits of each of our design choices by performing an ablation study. In future work, we would like to incorporate better modeling of latent dialog frames so as to improve the attention signal on our multi-level memory. As our error analysis suggests, nearly 22% of the errors could possibly be reduced by improved modeling of the dialog context. We believe that model performance can also be improved by capturing user intent better in case of non-sequential dialog flow.

References

Antoine Bordes and Jason Weston. 2017a. Learning end-to-end goal-oriented dialog. *ICLR*, abs/1605.07683.

Antoine Bordes and Jason Weston. 2017b. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*.

Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauro, and Yoshua Bengio. 2016. Hierarchical memory networks. *CoRR*, abs/1605.07427.

Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, and Dawei Yin. 2018. Hierarchical variational memory network for dialogue generation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1653–1662, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: A corpus for adding memory to goal-oriented dialogue systems. *CoRR*, abs/1704.00057.

Mihail Eric and Christopher D Manning. 2017. Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

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

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1437–1447.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. 2018. Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems. *arXiv preprint arXiv:1804.08217*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. 2017. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Hannes Schulz, Jeremie Zumer, Layla El Asri, and Shikhar Sharma. 2017. A frame tracking model for memory-enhanced dialogue systems. *arXiv preprint arXiv:1706.01690*.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. 2017. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *arXiv preprint arXiv:1706.09799*.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562. ACM.

Jian Su, Xavier Carreras, and Kevin Duh, editors. 2016. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. The Association for Computational Linguistics.

Da Tang, Xiujun Li, Jianfeng Gao, Chong Wang, Lihong Li, and Tony Jebara. 2018. Subgoal discovery for hierarchical dialogue policy learning. *arXiv preprint arXiv:1804.07855*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the International Conference on Machine Learning*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449. Association for Computational Linguistics.

Jiaming Xu, Jing Shi, Yiqun Yao, Suncong Zheng, Bo Xu, and Bo Xu. 2016. Hierarchical memory networks for answer selection on unknown words. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2290–2299.