Improving Generative Dialogue Models by Modeling Structures in Conversations

Anonymous ACL submission

Abstract

Task-oriented dialogues involve exchange of information in natural language between a user and an agent towards achieving a specific goal. These conversations are governed by observable information flow in the form of slot-value pairs and other latent structures. Generative sequence-2-sequence (Seq2Seq) models seldom leverage the structural information in response generation. To make sure the encoder representations are informative enough for the decoder, we propose and analyse two models: First, *Memory Palace* – addresses the encoder representation as a set of discrete blocks that are attended with a self-attention mechanism during inference. The inductive bias of the model forces an information bottleneck that helps learn dependency structures in the encoder latent representation. Second, *StructGen* – predicts the slot-value information as an auxiliary task for next utterance generation. Solving the auxiliary task makes the encoder representation to retain the information in long conversational contexts. We experiment the proposed models on Frames and MultiWoZ 2.0 datasets for next utterance generation task. The results show that both the models perform better than the baselines; and StructGen performs the best in the two datasets.

1 Introduction

Understanding the structure in inputs is challenging yet crucial to predict output in large spaces. Many natural language tasks have inherent structures – like parts of speech structure (Zeman et al., 2018), semantic structure (Banarescu et al., 2013; Williams et al., 2017). Dialogue tasks in goal-oriented conversations not only have such sentence level structures locally but also an higher level structure established through the exchange of information that is not restricted to the slot-value pairs (Serban et al., 2016).

USER:I am in town center and I need somewhere to go.
AGENT: There are 44 attractions near you, can you narrow down what you would like to see ?
USER: Anything is fine. Send me to your favorite place in the city, just be sure to give me the postcode.
AGENT: The man on the Moon concert hall is my favorite. Their postcode is cb12lf. Can I help you with anything else today ?
USER:I am looking for a train from Ely to Cambridge.
AGENT: What day would you like to travel, and at what time ?
USER: I'm going Monday and need to leave after 08:00.
AGENT: There are 8 trains. TR2987 leaves are 09:35 if you'd like that.
USER: Thanks. I will go ahead and book later. I think that's all the information I need for today.
AGENT: All right, then. Thank you for calling.

Figure 1: An example task-oriented conversation where an user intends to look for places to visit and get information about booking a train. The agent asks (slots) for specific information (time, etc.,) to which the user provides a specific response (value). A dialogue state can be defined as a summary of slots and values exchanged in the conversation so far to help continue the conversation.

Dialogue state tracking (Williams et al., 2013) is an active area of research where the objective is to predict the current state of the dialogue to be used for response selection in a discriminative way. Whereas, generative dialogue models encode the state information implicitly in the encoder summary that is used by the decoder for generating an appropriate response. In task-oriented dialogues, understanding the information flow through the slots and values shared by the user that make up the purpose of the conversation is important. Neural network models model the dialogue task into conditional language generation task using encoder-decoder style neural-network architectures (Vinyals and Le, 2015; Sutskever et al., 2014). In modeling task-oriented dialogues, the encoder network summarizes the conversation so far in a vector that is processed by the decoder network to generate a response that is by an agent/machine. As some practical implementations for large scale problems have shown performance degradation that might be attributed to the limitations of these models in exploiting such structures (Serban et al., 2017; Gontier et al., 2018), it is not clear if generative dialogue models learn to condition the responses on the right set of information shared by a user.

In this work we propose two generative approaches to leverage the information and structure in the input to improve inference time performance. First, we propose and analyze a weakly supervised module – *Memory Palace* – that has an inductive bias to read different non-contiguous units in encoder representation. The model has a *reader module* with multi-head self-attention mechanism allowing the reader module to read different units of encoder representation and selectively pass them through to the decoder. This helps the encoder representation to have different blocks in its representation to be useful for different context. We show that Memory Palace module can be used together with other encoder-decoder architectures as base models. In our experiments, we observed that when combined with a base model, the Memory Palace module improves model performance.

To make use of datasets that have annotations for the slot-value pairs and to compare Memory Palace architecture, we propose our second model – *StructGen*. StructGen trains an auxiliary network to predict the current dialogue state using the encoder representation of the conversation history. StructGen expects slot-value labels to train the auxiliary network. We observed in our experiments that predicting the dialogue state as an auxiliary task improves model performance significantly better than other sequence-2-sequence architectures.

The contributions of the paper are summarized as follows:

- We propose, Memory Palace, a weakly supervised architecture that learns a dependency structure between its discretely addressable memory blocks.
- We propose another architecture, StructGen, a sequence-2-sequence model with an auxiliary network that predicts the sets of slots and values exchanged until in a conversation.
- We show improvements in MultiWoZ 2.0 and Frames datasets by leveraging on the conversational strucuture.

2 Background

A task-oriented dialogue is an interaction between an user and an agent towards solving a one or more specific tasks through conversation. A task-oriented dialogue is represented as an alternating sequence of utterances $\{(u_1,a_1),(u_2,a_2),\dots,(u_n,a_n)\}$, where each u_i or a_i is a sequence of words, $w_1^{u_i}, w_2^{u_i}, \dots, w_n^{u_i}$. Building a generative dialogue model requires learning to generate a_t conditioned on the conversation history until as shown in Equation 1,

$$P(a_t \mid H_t) = \prod_{i=1}^{k} P(w_i^{a_t} \mid w_{<i}^{a_t}, (u,a)_{1-t}) \quad (1)$$

where H_t is the set of all previous interactions between the user and agent.

2.1 Sequence-to-Sequence Models

A basic sequence-to-sequence model (Vinyals and Le, 2015) consists of a recurrent neural network (RNN) (Lang et al., 1990) encoder followed by a RNN decoder. The encoder takes one word at a time and encodes the entire sequence of sentences into a single dense vector. At any time step t, the RNN encoder updates its hidden state as a function of the current input $\mathbf{w}_t$ and the previous hidden state $\mathbf{h}_{t-1}^{enc}$:

$$\mathbf{h}_t^{enc} = f(\mathbf{w}_t, \mathbf{h}_{t-1}^{enc}) \quad (2)$$

where f can be a RNN (Bengio et al., 1994) or an LSTM (Hochreiter and Schmidhuber, 1997) or a GRU (Cho et al., 2014) *cell*. The final hidden state of the decoder (say $\mathbf{h}_T^{enc}$ where T is the total number of words in the context) represents a single vector summary of the context. This vector is given as input to the decoder RNN at every step of decoding. The decoder RNN at any time step t first updates its hidden state as a function of the previously generated word $\mathbf{w}_{t-1}$, previous hidden state $\mathbf{h}_{t-1}^{dec}$, and the encoder output $\mathbf{h}_T^{enc}$:

$$\mathbf{h}_t^{dec} = f(\mathbf{w}_{t-1}, \mathbf{h}_{t-1}^{dec}, \mathbf{h}_T^{enc}) \quad (3)$$

where f is an RNN; that can instead be replaced with an LSTM or a GRU. Given $\mathbf{h}_{t-1}^{dec}$, the next word is predicted as follows:

$$\mathbf{w}_t = g(\mathbf{h}_t), \quad (4)$$

where g is a linear layer followed by a softmax non-linearity. The model is trained by maximizing the log-likelihood of the output sentence given the input sentences. For the decoder RNN, we can either feed the ground truth $\mathbf{w}_{t-1}$ while training (Williams and Zipser, 1989) or we can also periodically feed the model predicted $\mathbf{w}_{t-1}$ with a coin toss (also known as scheduled sampling (Bengio et al., 2015)). Scheduled sampling is shown to be better in training RNNs since it is difficult for teacher-forced RNNs to generalize when the previous predictions go wrong.

Hierarchical Recurrent Encoder-Decoder (HRED) (Sordoni et al., 2015) model has an encoder to encode the word level sequence in sentence, and another recurrent encoder to encode the sequence of encoded sentences. This sequence of encoded sentence vectors $\mathbf{h}_{S_1}^{w-enc}, \mathbf{h}_{S_2}^{w-enc}, \ldots, \mathbf{h}_{S_T}^{w-enc}$ from the word level encoder is given as input to the sentence-level RNN encoder to produce encoding for the sequence of sentences: $\mathbf{h}_T^{s-enc}$. $\mathbf{h}_T^{s-enc}$ is used by a word-level decoder, like in the basic sequence-to-sequence architecture, to generate the output sequence.

2.2 REINFORCE

Sampling from a model distribution breaks the gradient propagation to the architecture. REINFORCE (Williams, 1992) is a gradient update technique used to update the parameters when the loss surface is not differentiable.

$$\nabla_\theta \mathbb{L}(\theta) = \nabla_\theta \mathbb{E}_\theta (R) \tag{5}$$

$$= \sum_{a_k} \pi_\theta (a_k \mid s_k) \nabla_\theta \log \pi_\theta (a_k \mid s_k) R \tag{6}$$

The update equation is as follows:

$$= \mathbb{E}_\theta [\nabla_\theta \log \pi_\theta (a_k \mid s_k) R] \tag{7}$$

REINFORCE assumes a differentiable parameter space, $\theta \sim \Theta$, that predicts a *policy* π_θ conditioned on an input, $\theta : \mathbf{s}_k \rightarrow \pi_\theta$. When choosing a sequence of actions, a, sampled from π_θ results in a cumulative return, R, the parameter updates are made as shown in Equations 5-7.

2.3 Evaluation Metrics

We evaluate the performance of the models proposed in this paper against the baselines using the metrics discussed in this section.

2.3.1 BLEU-N Score

BLEU-*N* (Papineni et al., 2002) is an overlap based metric computed by analyzing the co-occurances of *N*-grams. The score is computed with an n-gram precision (P_n) computed over the sequence. To penalize shorter sentences the score uses a brevity penalty.

Let r and $\hat{r}$ be the target and model predicted sentences,

$$\text{BLEU-N} = b(r, \hat{r}) \exp \sum_{n=1}^{N} \beta_n \log P_n(r, \hat{r}) \tag{8}$$

$b(\cdot)$ is the brevity penalty to penalize shorter responses from model.

2.3.2 ROUGE Score

ROUGE (Lin, 2004) score is a set of overlap based metrics that evaluates using the amount of word overlap between a source and a target text. We use ROUGE-L that computes f-measure based on the longest common subsequence (LCS) between the source and the target. There can be other words in the LCS unlike an n-gram overlap. This is better than the other overlap based metrics by providing a measure on precision and recall that also evaluates the syntactic similarities between the sentences.

3 Memory Palace

We propose an architecture, *Memory Palace* (Figure 2), with inductive bias to learn dependencies in the latent representation of the encoder. The encoder summarizes the conversation history that has defined relations (slots) and entities (values) – the encoder summary of the utterances. The slots-values comprise of the structure that is observable in the annotations of the conversations. There could be other implicit structures in the language than the slot-value information used for tracking the conversation. Memory Palace uses such implicit and observable structures in dialogue generation with self-attention over the encoder representation.

Memory Palace architecture has three components: an *encoder module*, a *reader module*, and a *decoder module*.

3.1 Encoder Module

The encoder module, a 2-layer LSTM, summarizes a subsequence of tokens in the input context. Although the reader module addresses the

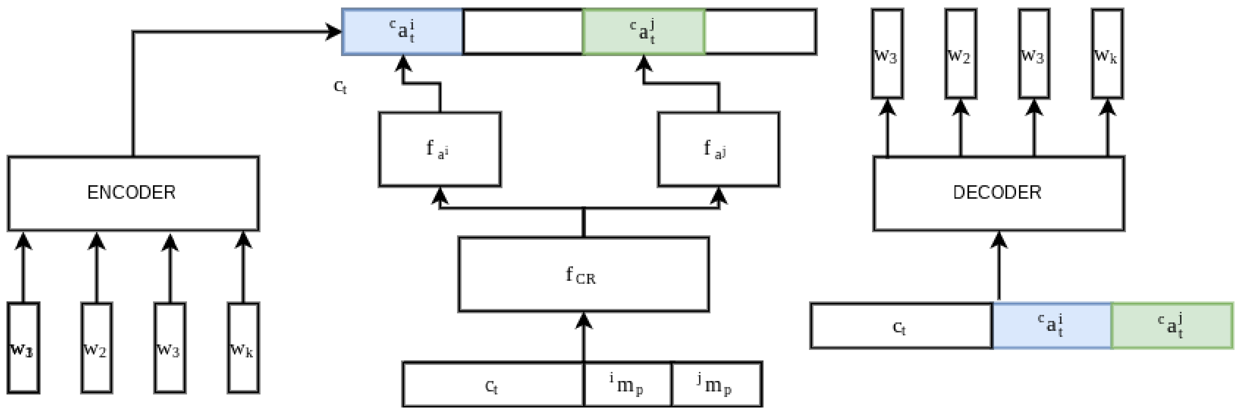

Figure 2: An illustration of the Memory Palace architecture with the encoder, reader and decoder modules. The reader module in the illustration has 2 read heads that copies the content from specific blocks in the encoder representation.

encoder representation as individual blocks, the blocks are not individually written like in memory augmented architectures.

The model does not have a separate memory structure but maintains the encoded summary vector to be addressable like a lookup table. The reader module looks up for discrete blocks of content from the summary vector and backpropagates through those units.

3.2 Decoder Module

The decoder module is a 2-layer LSTM that is trained with scheduled sampling to predict the response corresponding to an input context. This module takes in as input the contents of different memory blocks and a summary of the context to generate a natural language text sequence as a response to the input context. The working of the decoder module and the reader module is analogous to decoding a graph (Simonovsky and Komodakis, 2018; You et al., 2018), where the memory blocks attended by the reader module are entities and the relationship between the entities is inferred from the summarized context. The model trains on minimizing the loss in Equation 9.

$$\mathbb{L}^r = -\sum_{i=1}^{T} \log P\left(w_i \mid \hat{\mathbf{c}}_t, \left({}^{c}\mathbf{a}_t^k\right)_{k=1}^{K}\right) \quad (9)$$

where $\left({}^{c}\mathbf{a}_t^k\right)_{k=1}^{K}$ denote the contents selected by the K read heads in the reader module (Section 3.3).

3.3 Reader Module

The reader module is a Fully-Connected neural (f_{CR}) network that predicts a common representation. The common representation is used as input by K other fully-connected networks (f_{a^k}), where K is the number of attention heads. A memory block (m_i) from the set of all memory blocks in the encoder representation, M, is selected with a softmax operation over f_{a^k} . The selection of a memory block(i) attended by attention head k is selected as shown in Equation 10.

$${}^{k}m_i = \mathtt{argmax}_p\ \mathtt{softmax}(f_{a^k}(f_{CR}(\mathbf{c}_t, ({}^{k}m_p)_{k=1}^{K}))) \quad (10)$$

where ${}^{k}m_p$ is a probability distribution over memory blocks. ${}^{k}m_i$ is expanded into a mask over $\mathbf{c}_t$ over the selected block and for the size of the block, b. The contents of the blocks are copied with an element-wise dot product, as shown in Equation 11.

$${}^{c}\mathbf{a}_t^k \leftarrow {}^{k}\, m_i \cdot \mathbf{c}_t \quad (11)$$

The size of the memory blocks, the number of heads and the amount of overlap between two adjacent blocks are hyperparameters of the model. The hard-attention mechanism updates the parameters of the encoder and self using REINFORCE (Equation 7) where the reward in the Equation 15. The *action* for the reader module is to select the memory blocks indices that has information necessary to predict the next utterance. Hence, the reader module receives the negative log-likelihood loss as the cumulative reward over the prediction of sequence of tokens.

The hard-attention mechanism updates only the encoder weight parameters of the attended blocks. This forces the reader module and the encoder to learn a representation indicating relationships between the different memory blocks. The reader module uses that in improving the next utterance prediction.

4 StructGen Architecture

StructGen is an extension to Sequence-2-Sequence architecture that trains an auxiliary network to predict the slot-value pairs that is provided in most of the task-oriented dialogue datasets (Budzianowski et al., 2018; Asri et al., 2017; Williams et al., 2013) used in the literature. The conversation history between an user and the agent is a long sequence of words with sparse information. Without necessary supervision on the encoding of the conversation, the probability of committing errors that arise from insufficient information provided to the decoder from the encoder increases. *StructGen* architecture (Figure 3) trains an auxiliary network that is trained with supervision to predict the current dialogue state comprising of slots and values. This assists in keeping track of the information exchanged.

The auxiliary task loss helps encoding the current dialogue state information in the encoder representation which helps improving the next utterance prediction by the decoder. As next utterance prediction requires the model to have a good understanding of the dialogue state, the auxiliary loss aids the encoder representation to improve the performance of the model.

StructGen architecture has (1) Input Encoder, (2) Slot-Value Predictor, (3) Output Decoder. We discuss the components in detail further in this section.

4.1 Input Encoder

Input Encoder module is a 2-layer LSTM that encodes the input context from the user. The Input Encoder encodes the last 100 tokens and maintains the window size constant. The sequence of tokens are encoded to a d-dimensional vector encoding $\mathbf{c}_t$, where d is the size of the LSTM hidden state.

4.2 Slot-Value Predictor

The input context for an agent to respond is a natural language encoding of information from user. Every sentence in the context is a natural language expression of a set of slots: (sl_i, vl_i) where sl_i denote the slots and vl_i is the value provided by the user for the corresponding slots. Slot-Value Predictor network takes as input the encoded context summary, $\mathbf{c}_t$. The Slot-Value Predictor module has two fully-connected networks as shown in Equation 12 and 13. The two networks predict the slots and values in the input context that are valid until the conversation so far.

$$sl_t = f_{slot}(\mathbf{c}_t) \tag{12}$$

$$vl_t = f_{value}(\mathbf{c}_t) \tag{13}$$

$\mathbf{c}_t$ is conditioned on to train two fully connected networks that independently predicts slots and values in the encoded conversation history.

$$\mathbb{L}^f = \mathbb{H}(sl_t, sl_t^*) + \mathbb{H}(vl_t, vl_t^*) \tag{14}$$

where $\mathbb{H}(\cdot)$ is the cross-entropy loss and sl_t^* and vl_t^* are ground truth labels for slots and values.

The input encoder receives scalar feedback from the slot-value prediction loss and the language model loss; the slot-value predictor networks optimize on the loss in Equation 14 to predict the right slot-value pairs in the encoded context.

4.3 Output Decoder

The *Output Decoder* module is a 2-layer LSTM which is trained to predict the sequence one token at a step by conditioning on all previous generated tokens and the encoded context, $\mathbf{c}_t$. This response decoder RNN is trained by maximizing the conditional log-likelihood as shown in Equation 15. The decoder module is trained with Scheduled Sampling.

$$\mathbb{L}^r = -\sum_{i=1}^{T} \log P(w_i \mid \hat{\mathbf{c}}_t) \tag{15}$$

A linear combination of the two loss functions (Equation 14, 15) as shown in Equation 16 is used as the optimization criterion for training.

$$\mathbb{L}_{train} = \alpha * \mathbb{L}^f + (1-\alpha) * \mathbb{L}^r \tag{16}$$

The auxiliary local graph prediction serves two purposes: Firstly, it provides an interpretable summary of the maximum-likelihood estimate of the agent's current dialogue state. Secondly, the auxiliary loss improves the discriminative capability of the encoder representation by regularizing the representation by using it to solve more than one task.

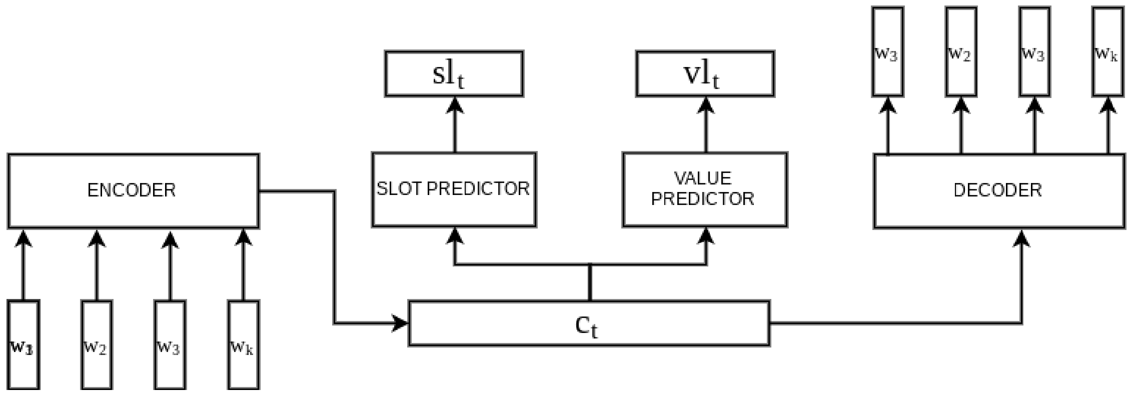

Figure 3: The StructGen architecture with *encoder module*, *decoder module* and *slot-value predictor*. The slot predictor and value predictor networks' prediction can be used to represent the local dialogue graph.

5 Experiments

The experiments were set up to understand the effects of auxiliary task to predict the local graph from the context – in StructGen – and the weak supervision through REINFORCE in Seq2Seq and HRED models combined with Memory Palace. Further, we analyze the effects of overlap in Memory Palace combined encoder-decoder architectures. As the experiments relied on natural strucuture in conversations, we use goal oriented dialogue datasets – Frames and MultiWoZ 2.0 to train and evaluate our models. We use automatic evaluation metrics – ROUGE-F1 and METEOR – as a proxy to human evaluation on the task-oriented datasets (Sharma et al., 2017).

5.1 Datasets

The information in goal-oriented conversation is rich and models benefit in generation tasks by exploiting the structures. Frames (Asri et al., 2017) is a restaurant booking dataset that provides information about the dialogue state with the *frames*. This has speaker information, entities, relations and information provided by the user for a slot. The dataset has around 1200 dialogues for training 80 dialogues to test. The MultiWoZ dialogue dataset is a collection of human-to-human task-oriented interactions over multiple domains (Budzianowski et al., 2018). The conversations involve a visitor and a clerk from an information centre point. The human performing the role of a visitor is provided with a template for the conversation that defines a set of instructions needed to be fulfilled. The dataset is currently the largest publicly available goal oriented dialogue dataset with $\sim$ 8400 dialogues to train and 1000 dialogues to test. The vocabulary in the dataset spans over multiple domains making it one of the challenging datasets.

5.2 Results

The results combining the best performing StructGen architecture and Memory Palace with Seq2Seq and HRED are shown in Tables 1 and 2. The results show that StructGen performed significantly better than the other models by learning to predict the observable structure provided in the slot-value information as an auxiliary task. The improvement in performance by using auxiliary loss shows that auxiliary tasks could help retaining information when encoding longer sequences.

Although the Memory palace model learns hidden structures in the latent representations, we observed the learned structures to moderately overfit to the training set. The results in Tables 1 and 2 show that exploiting the structures using Memory Palace is much better than implicit encoding in the baseline models. Although the results of the experiments are limited to goal oriented dialogues, Memory Palace can be applied to any sequential data that involves interactions among entities in the data.

For the experiments, we used Memory Palace architecture with basic Seq2Seq architecture and HRED architecture. The best performing Memory Palace architecture on Frames dataset is when coupled with HRED; and the Seq2Seq + Memory Palace performed better than the baselines and HRED + Memory Palace in MultiWoZ dataset. Although the best performing combination differed in the two datasets, the models performed significantly poorer when not coupled with Memory Palace. This shows that the inductive bias of learning dependency structures in the latent representation helps in improving language generation. We observed that the combination of HRED and Memory Palace in large dataset to slightly affect the performance. This could be caused by difficulty in extracting relations from approximate higher level structures that HRED computes over its input.

Model	BLEU	ROUGE-F1
Seq2Seq	8.73	25.13
HRED	8.07	21.47
Seq2Seq + MP	11.74	27.01
HRED + MP	13.12	28.44
StructGen	**18.58**	**35.18**

Table 1: Comparison of the performance of different architectures against baseline models. The table shows the performance of models on Frames dialogue dataset. (MP stands for Memory Palace)

Though the performance could not beat that of StructGen's, the results show that inducing a structure with differential updating of encoder representation is useful than implicit encoding as in the baseline. Also, we observed in all the experiments that the reader module of Memory Palace predicted different locations for different contexts. The pattern, though, was not intuitive to understand.

Model	BLEU	ROUGE-F1
Seq2Seq	19.58	38.21
HRED	19.64	38.65
Seq2Seq + MP	20.56	38.33
HRED + MP	18.26	35.57
StructGen	**26.23**	**44.98**

Table 2: Comparison of the performance of different architectures against baseline models. The table shows the performance of models on MultiWoZ 2.0 dialogue dataset. (MP stands for Memory Palace)

5.2.1 Varying the overlap

In Memory Palace architecture, the reading of the encoder blocks is done by a multi-head self attention mechanism that copies the contents of the blocks. In addition to the language-model loss optimized by the model, the behavior policy (reinforce) learnt by the read mechanism to copy the different contents of the memory is also optimized. The read mechanism forces an information bottleneck between the encoder and the decoder enabling the representation to be discriminatory. This helps the model have an improved performance. In the experiment in Table 3, we experiment with different amounts of overlap between two adjacent memory blocks by selecting the best performing hyperparameters for block size and number of read heads. In experiments on MultiWoZ, the best hyperparameter for number of read heads and block size are: 3 and 16, in Frames: 2 and 8. The encoder hidden sizes were 128 and 256 for Frames and MultiWoZ respectively.

Overlap	BLEU	ROUGE-F1
0	10.09	26.61
1	**13.12**	**28.44**
2	10.03	24.89
4	10.84	25.26
6	9.42	23.87

Table 3: Comparison of HRED + Memory Palace with different amount of overlap between adjacent memory blocks. The model is experimented with hyperparameters hidden_size 128, block_size 8 and number of heads 2.

We observed from experiments on Seq2Seq + Memory Palace on MultiWoZ dataset that the minimal correlation in updates favors better resolution of information in the encoder representation. The Memory Palace architecture favors memory blocks with little to no overlap. This is could be explained as that is when the effects of differential updates is effective. We observed similar results in experimenting with different amounts of overlap with HRED + Memory palace in Frames dataset (Table 3).

The improvement in performance over the baseline models could be attributed to the baseline models' maintaining of a single large encoding. Such a representation can overfit to generate the most-likely due to the lack of discriminative evidence in the latent space. Memory palace handles this with differential updates whereas StructGen by regularizing the representation with an auxiliary task.

Overlap	BLEU	ROUGE-F1
0	**20.56**	**38.33**
1	18.31	35.52
2	18.32	35.05
4	18.5	36
6	18.4	35.51

Table 4: Comparison of Seq2Seq + Memory Palace with different amount of overlap between adjacent memory blocks. The model is experimented with hyperparameters hidden_size 256, block_size 16 and number of heads 3.

6 Related Work

The encoder-decoder style models were used for dialogue tasks (Vinyals and Le, 2015). This, as mentioned in Section 2, considers the text as a sequence of words without considering the hierarchical structures. To capture the higher level sequential structures (Sordoni et al., 2015; Serban et al., 2015, 2016) proposed hierarchical encoders that stacks two RNNs to capture hierarchical information as contained at different levels of the conversation. Despite these models attempting to capture the dialogue states in a generative modeling set up, the models have struggled in responding coherently that is probably caused by lack of sufficient dialogue state information.

Dialogue state tracking (Williams et al., 2013) is one of the core components of a dialogue system that requires keeping track of the information exchanged by an user to the agent. Several works in this area of research have looked at predicting the state in goal-oriented dialogues through discriminative (Mrkšić et al., 2016; Ultes et al., 2017) or generative approach (Neelakantan et al., 2019) by computing belief over the values for different slots exchanged. StructGen shares similarity to some of the approaches in dialogue state tracking but differs in the way the state prediction is set up as an auxiliary task for language generation.

To some extent, the difficulties in caturing dialogue state is overcome with supplementing an external knowledge, either as a vector encoding or queried from external sources of knowledge as shown in (Wen et al., 2017; Guo et al., 2018; Parthasarathi and Pineau, 2018). Such models query for external knowledge sources during inference time and augment the results to the decoder for improving the next utterance prediction. The shortcoming of such approaches is that the usefulness of the information queried or stored is independent of the task thus making the querying a huge overhead.

Weston et al. (2014, 2015) assume the existence of memory created *apriori* with the necessary facts that allows a model to select the required facts during the course of conversation. These models with external memory networks require labeled data to store entity relation information that can be used during inference. Further, to allow sequence modeling tools to store long range dependencies, usage of external memory component have been experimented on toy tasks. Such models construct a modularized memory component. Models like Neural Turing Machine (Graves et al., 2014; Gulcehre et al., 2016), TARDIS (Gulcehre et al., 2017) or DNC (Graves et al., 2016) train a controller unit that learns memory manipulation functions including attention, read and write. Conceptually such external memory models allow the model to learn over long sequences; but they are computationally slow and are difficult to train.

7 Conclusions and Future Work

The paper proposes two novel sequence-to-sequence architectures for dialogue generation that make use of the conversational structure. The first model, Memory Palace, learns dependencies between independent memory blocks in the latent representation for language generation whereas the second model StructGen – predicts the current dialogue state of slot-value pairs as an auxiliary task. The results showed that the Memory Palace architecture performs better than the baselines and can be used for language generation tasks in datasets with no annotations on the slot-values or other observable structures. The significantly better performance of the supervised StructGen approach shows that explicit validation of the encoder representation for the existence of relevant information helps in improving the language generation. The only limitation of this approach though is requiring the annotated labels.

References

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. *arXiv*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Proceedings of NeurIPS*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, and Milica Gašić. 2018. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of EMNLP*.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *arXiv*.

Nicolas Gontier, Koustuv Sinha, Peter Henderson, Iulian Serban, Michael Noseworthy, Prasanna Parthasarathi, and Joelle Pineau. 2018. The rllchatbot: a solution to the convai challenge. *arXiv preprint*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv*.

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*.

Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections. *arXiv*.

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. 2016. Dynamic neural turing machine with soft and hard addressing schemes. *arXiv*.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Proceedings of NeurIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*.

Kevin J Lang, Alex H Waibel, and Geoffrey E Hinton. 1990. A time-delay neural network architecture for isolated word recognition. In *Neural networks*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv*.

Arvind Neelakantan, Semih Yavuz, Sharan Narang, Vishaal Prasad, Ben Goodrich, Daniel Duckworth, Chinnadhurai Sankar, and Xifeng Yan. 2019. Neural assistant: Joint action prediction, response generation, and latent knowledge reasoning. *arXiv preprint*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Prasanna Parthasarathi and Joelle Pineau. 2018. Extending neural generative conversational model using external knowledge sources. In *Proceedings of EMNLP*.

Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. 2017. A deep reinforcement learning chatbot. *arXiv*.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *arXiv*.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. A hierarchical latent variable encoder-decoder model for generating dialogues. In *arXiv*.

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*.

Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv*.

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 International on Conference on Information and Knowledge Management*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *arXiv*.

Stefan Ultes, Lina M Rojas Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Inigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gasic, et al. 2017. Pydial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL 2017, System Demonstrations*.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gašić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. *Proceedings of EACL*.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv*.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv*.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv*.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*.

Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. 2018. Graphrnn: A deep generative model for graphs. *arXiv*.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 2018. Conll 2018 shared task: multilingual parsing from raw text to universal dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.