Retrieve & Memorize: Dialog Policy Learning with Multi-Action Memory

Yunhao Li[1], Yunyi Yang[1], Xiaojun Quan[1]*, Jianxing Yu[2]
[1]School of Computer Science and Engineering, Sun Yat-sen University, China
[2]School of Artificial Intelligence, Sun Yat-sen University, China
{liyh355,yangyy37}@mail2.sysu.edu.cn
{quanxj3,yujx26}@mail.sysu.edu.cn

Abstract

Dialogue policy learning, a subtask that determines the content of system response generation and then the degree of task completion, is essential for task-oriented dialogue systems. However, the unbalanced distribution of system actions in dialogue datasets often causes difficulty in learning to generate desired actions and responses. In this paper, we propose a retrieve-and-memorize framework to enhance the learning of system actions. Specially, we first design a neural context-aware retrieval module to retrieve multiple candidate system actions from the training set given a dialogue context. Then, we propose a memory-augmented multi-decoder network to generate the system actions conditioned on the candidate actions, which allows the network to adaptively select key information in the candidate actions and ignore noises. We conduct experiments on the large-scale multi-domain task-oriented dialogue dataset MultiWOZ 2.0 and MultiWOZ 2.1. Experimental results show that our method achieves competitive performance among several state-of-the-art models in the context-to-response generation task.

1 Introduction

Task-oriented dialogue systems communicate with users through natural language conversations to accomplish a wide range of tasks such as restaurant and flight bookings. Recent years have seen a rapid growth of interest in building task-oriented dialogue systems (Budzianowski et al., 2018). Such systems are usually decomposed into several subtasks, including natural language understanding (Gupta et al., 2018), dialogue state tracking (Zhong et al., 2018), system actions (dialogue policy) prediction, and response generation (Wen et al., 2015; Chen et al., 2019; Zhao et al., 2019), where system actions can be viewed as a semantic plan of response generation. One of the main challenges for context-to-response generation in task-oriented dialogue systems comes from the intrinsic one-to-many property in conversations. As shown in Figure 1, there can be multiple valid system actions for the same dialogue context, which means that multiple satisfactory system responses can be generated correspondingly. However, in most collected dialogue datasets, each dialogue context has only one reference, which leads to an unbalanced distribution of system actions and responses in multi-domain dialogue datasets (Zhang et al., 2020). Models trained on such unbalanced datasets tend to overfit high-frequency system actions and underfit low-frequency ones.

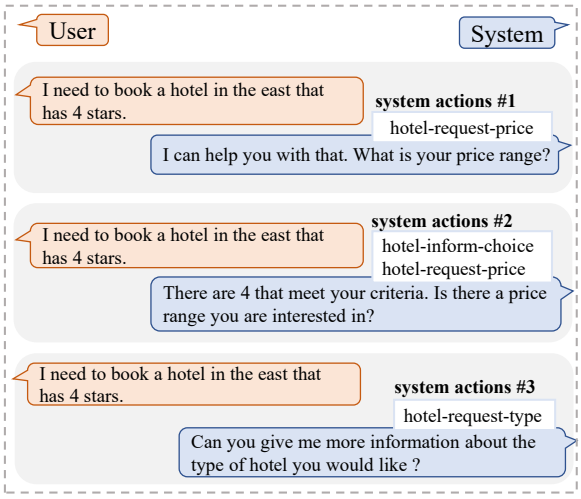

Figure 1: An example of the one-to-many property,where there are multiple appropriate system actions and responses given the same dialogue context.

One line of work focuses on the representation of system actions, which alleviates the unbalanced problem to a certain extent. Chen et al. (2019) reconstruct system actions into a compact graph representation. Zhao et al. (2019) treat system actions as latent variables and use reinforcement learning

* Corresponding author

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 447–459

to optimize them. Wang et al. (2020b) model system actions prediction as a sequence generation problem by treating system actions as a sequence of tokens. On the other hand, Zhang et al. (2020) explicitly modeling the one-to-many property to enrich system action diversity through a rule-based multi-action data augmentation. Specifically, they treat system actions that follow the same dialogue state as alternative valid actions and train them together with the reference system action. However, their data augmentation framework has two shortcomings. First, it enforces a rigid mapping between dialogue state and system actions. Dialogue state, which consists of information such as belief state and user actions, is not flexible enough to represent the whole dialogue context and thus limits the diversity of the mapped system actions. Second, they treat the mapped system actions as gold references during training which may force the model to fit noise in the mapped system actions and ultimately hinder the quality of the generated system actions.

To address the above limitations, we propose to model the one-to-many property more effectively by retrieving multiple candidate system actions and selectively taking the candidates into consideration when generating system action. We design a retrieve-and-memorize framework that consists of a context-aware neural retrieval module (CARM) and a memory-augmented multi-decoder network (MAMD). Specifically, the context-aware retrieval module uses a pre-trained language model to convert the dialogue history as well as belief state into a context representation of each sample. Multiple candidate system actions are retrieved based on the distances between the context vector and the representations of other samples in the latent space. These retrieved candidate actions are more diverse and consistent with the dialogue context since they are obtained based on a more holistic representation. Instead of treating the candidates impartially with the gold references, we encode them into a memory bank and the memory-augmented multi-decoder network can dynamically attend to the memory bank during system actions generation. Additionally, we employ a random sampling mechanism where during training, the memory bank is filled with randomly sampled system actions with a probability, which allows the model to learn to distinguish the quality of the candidates and adaptively adjust its dependence on the candidate actions.

We evaluate our model on MultiWOZ (Budzianowski et al., 2018), a large-scale multi-domain dataset for task-oriented dialogue systems. Extensive experiments and analyses are conducted to demonstrate the effectiveness of our model, and the results show that it significantly outperforms the baseline model. Our main contributions are summarized as follows:

- We propose a context-aware retrieval module that can retrieve multiple appropriate system actions given a dialogue context.
- We propose a memory-augmented multi-decoder network that can generate system actions based on multiple candidate actions.
- Our model outperforms several state-of-the-art baselines on a large-scale multi-domain dataset for task-oriented dialogue systems.

2 Related Work

One line of research focuses on the representation of system actions. A typical approach to encoding system actions is by concatenating the one-hot representation at each level of actions into a flat vector (Wen et al., 2015; Budzianowski et al., 2018). Such sparse representations make the learning of system actions difficult. To overcome the sparsity issue, Chen et al. (2019) compact the one-hot vector representation based on the intrinsic hierarchical structures of system actions, and apply hierarchical disentangled self-attention to generate system response. Zhao et al. (2019) treat system actions as latent variables and use reinforced learning (He et al., 2016) to optimize them. Recently, Wang et al. (2020b) propose a co-generation framework to generate system actions and response sequentially, which achieves a new state of the art in the context-to-response task. Our proposed framework adopts the idea of modeling belief state and system actions (Wang et al., 2020b; Liang et al., 2020) as sequences and generates the belief state, system action, and response sequentially to make better use of the intermediate supervision.

Another line of research uses data augmentation to expand the training data. Gao et al. (2020) use the paraphrase technique (Li et al., 2019; Wang et al., 2019) to generate user utterances and then expand the training set with the augmented user utterances. Zhang et al. (2020) augment system actions with a mapped dialogue state, which consists of belief state, user action, turn domain, and

database search result. Such mapping is rule-based and requires user actions for the construction of dialogue state, which takes extra annotations. Both of the above approaches treat the augmented samples as equivalent to the gold ones, which may force the model to fit noises in the augmented data. In this paper, we focus on a better neural retrieval method for the alternative system actions, and instead of directly training on the augmented actions, we encode them in a memory bank as auxiliary information.

3 Methodology

To frame the problem of dialogue policy learning, we use $X_t = \{U_1, .., U_{t-1}, R_{t-1}, U_t\}$ to denote the dialogue history at turn t of a multi-turn conversation, where $U_i = u_1 u_2, ...u_{m_i}$ and $R_i = r_1 r_2 ... r_{n_i}$ are the i-th user utterance and system response, respectively. Following previous works (Zhang et al., 2020; Liang et al., 2020), we convert the belief state and system actions from a list of triples to sequences. For example, the belief state "*restaurant-food-Chinese,restaurant-price-expansive*" is converted to "*restaurant [food] Chinese [price] expansive*", and the system actions "*restaurant-inform-price,restaurant-inform-phone*" are converted to "*restaurant [inform] price phone*". We use $B_t = b_1 b_2 ... b_p$ and $A_t = a_1 a_2 ... a_q$ to represent the current belief state and system action, respectively. Our goal is to generate system actions A_t and system response R_t of turn t based on the dialogue context X_t and belief state B_t.

We employ a retrieve-and-memorize framework to generate the system response. First, we use a context-aware retrieve module to retrieve multiple proper candidate system actions from the training set. Then, we encode the candidate actions into a memory bank and propose a memory-augmented module to enhance the action generation.

3.1 Context-Aware Retrieval Module

In order to retrieve alternative system actions that are more comprehensive and context-aware, we utilize the powerful pre-trained language model BERT (Devlin et al., 2019) to obtain distributed representations of the dialogue context. We search in the training corpus for system actions with similar distributed representations and retrieve them as alternative candidate actions. Concretely, we combine the dialogue history $X_t = \{U_1, .., U_{t-1}, R_{t-1}, U_t\}$

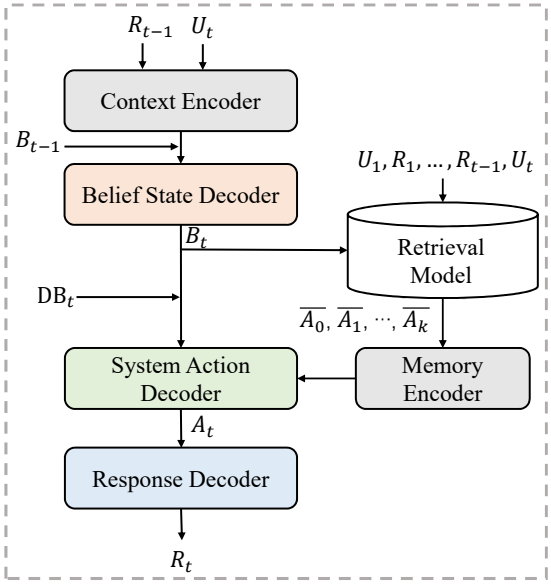

Figure 2: An overview of the proposed model.

and belief state B_t as dialogue context and feed the concatenated dialogue context into a pre-trained BERT encoder:

$$H = \text{BERT}([CLS] \oplus B_t \oplus [SEP] \oplus X_t), \quad (1)$$

where $\oplus$ is the concatenation operator, $[CLS]$ is a special token that precedes every input sequence of BERT, and $[SEP]$ is a special token used to separate different parts of the input sequence. The BERT model encodes the input dialogue context into a sequence of hidden states $H = \{h^{CLS}, h^1, ..., h^L\}$. We use h^{CLS} to represent the distributed representation of dialogue context, since h^{CLS} is expected to capture the information of the whole sequence. Then we use L_2 distance to measure the similarity between the distributed representations of different dialog contexts:

$$L_2(h_i^{CLS}, h_j^{CLS}) = ||h_i^{CLS} - h_j^{CLS}||_2. \quad (2)$$

Based on the L_2 distance, k most similar dialogue contexts are selected from the training set, and the corresponding system actions constitute a candidate actions set $\{\bar{A}_1, \bar{A}_2, ..., \bar{A}_k\}$.

Pre-training Task Directly applying h^{CLS} from BERT without fine-tuning or further pre-training may not result in desired dialogue context representations that correlate well with system actions. A good dialogue contextual representation should satisfy the property that dialogue contexts with similar semantics are close to each other in the representation space. Therefore, we further pre-train the

BERT model with an actions prediction task:

$$p(y|B_t, X_t) = \text{classifier}(h^{CLS}), \quad (3)$$

where $y \in \mathbb{R}^D$ is a one-hot label of system actions (Chen et al., 2019), D is the dimension of the label space, and *classifier* is a simple linear classifier.

3.2 Memory-Augmented Multi-Decoder Network

We propose a memory-augmented multi-decoder network that jointly generates belief state, system actions, and system response while having access to a memory bank when generating the system action. Given the retrieved candidate system actions, we encode these candidates into the memory bank and enhance the generation of system actions by querying the memory bank during decoding.

Encoding Module We use Bidirectional GRUs (Chung et al., 2014) as our encoders. First, we encode the current user utterance, the previous system response and the previous belief state separately into hidden states:

$$\begin{aligned} H_u &= \text{Encoder}(U_t), \\ H_{pre\_r} &= \text{Encoder}(R_{t-1}), \\ H_{pre\_b} &= \text{Encoder}(B_{t-1}), \end{aligned} \quad (4)$$

Then, another encoder is used to encode the candidate system actions into memory bank:

$$M_t = \text{Encoder}_M(\bar{A}_1 \oplus \bar{A}_2 \oplus ... \oplus \bar{A}_k), \quad (5)$$

where $M_t = \{m_1, ..., m_k\}$.

Belief State Generation The belief state B_t of turn t is generated based on the current user utterance U_t, previous system response R_{t-1} and previous belief state B_{t-1}. The generation of B_t at each time step τ can be formulated as follows:

$$\begin{aligned} s_\tau &= \text{Attn}(h_{\tau-1}, H_u, H_{pre\_r}, H_{pre\_b}), \\ c_\tau &= [s_\tau \oplus e(b_{\tau-1})], \\ p(b_\tau|b_{1:\tau-1}), h_\tau &= \text{Dec}_b(c_\tau, h_{\tau-1}, H_{pre\_b}), \end{aligned} \quad (6)$$

where $Attn$[1] is an attention function, $e(b_{\tau-1})$ is the embedding of the previous token, $h_{\tau-1}$ is the hidden state from the last decoding step, and $h_0 = \mathbf{0}$. Dec_b[1] is the belief state decoder augmented with copy mechanism (Gu et al., 2016), which can copy tokens from the previous belief state. $p(b_\tau|b_{1:\tau-1})$ is a distribution over vocabulary. We use cross entropy between ground truth and the output distribution $\mathcal{L}_b(\theta)$ as the loss of belief state generation. We collect the hidden states $H_b = \{h_0, h_1, ..., h_p\}$ of each step to feed them into the action decoder.

[1]Please refer to the appendix for more details.

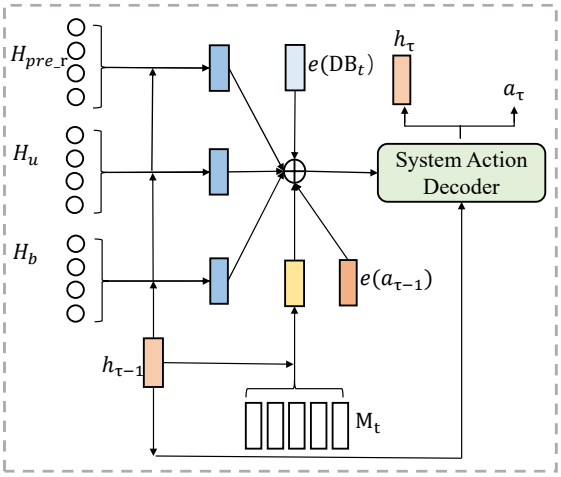

Figure 3: Generation of system actions at time step τ.

Memory-Augmented Action Generation As shown in Figure 3, the system action A_t of turn t is generated based on not only the dialog history and the current belief state, but also the memory bank which encodes the retrieved candidate system actions. For the generation of A_t, at each time step, we first compute the state s_τ:

$$s_\tau = \text{Attn}(h_{\tau-1}, H_u, H_{pre\_r}, H_b). \quad (7)$$

Then, we use the hidden state $h_{\tau-1}$ to query the encoded candidate system actions memory M_t:

$$\begin{aligned} a_\tau^i &= \tanh(W[h_{\tau-1} \oplus m_i]), \\ \alpha_\tau &= \text{Softmax}(a_\tau), \\ v_\tau &= \sum_{i=1}^{k} \alpha_\tau^i m_i, \end{aligned} \quad (8)$$

where W are learnable parameters and v_τ contains information from the memory. Now we incorporate v_τ into the generation process:

$$\begin{aligned} c_\tau &= [s_\tau \oplus e(a_{\tau-1}) \oplus e(DB_t) \oplus v_\tau], \\ p(a_\tau|a_{1:\tau-1}), h_\tau &= \text{Dec}_a(c_\tau, h_{\tau-1}, H_b), \end{aligned} \quad (9)$$

where $e(a_{\tau-1})$ is the embedding of the previous token, $e(DB_t)$ is the embedding of the database search result which indicates the number of matched entities. Dec_a is the action decoder augmented with copy mechanism. The cross entropy $\mathcal{L}_a(\theta)$ between the output distribution and ground truth is the loss of actions generation. We

collect the hidden states $H_a = \{h_0, h_1, ..., h_q\}$ as well to feed it into the system response decoder.

Random Sampling Though the retrieved candidate system actions are considered to be of high quality and suitable given the dialogue context, we would still like our model to avoid taking those candidates for granted and developing excessive dependence on them. To this end, during training, the memory bank is filled with randomly sampled system actions with a probability p, and retrieved candidates with a probability $(1 - p)$. This allows the model to learn to distinguish good candidates from bad candidates.

Response Generation Lastly, we generate the system response conditioned on the hidden states of user utterance H_u, belief state H_b and system actions H_a with the response decoder Dec_r:

$$\begin{aligned} s_\tau &= \text{Attn}(h_{\tau-1}, H_u, H_b, H_a), \\ c_\tau &= [s_\tau \oplus e(r_{\tau-1})], \\ p(r_\tau | r_{1:\tau-1}), h_\tau &= \text{Dec}_r(c_\tau, h_{\tau-1}, H_b), \end{aligned} \tag{10}$$

The response generation loss $\mathcal{L}_r(\theta)$ is the cross entropy between the output and ground truth.

Objective Function The final objective function is the sum of belief state loss, actions generation loss and response generation loss:

$$\mathcal{L}(\theta) = \mathcal{L}_b(\theta) + \mathcal{L}_a(\theta) + \mathcal{L}_r(\theta) \tag{11}$$

4 Experiments

4.1 Dataset and Metrics

We conduct our experiments primarily on MultiWOZ 2.0 (Budzianowski et al., 2018). It consists of 8438 dialogues spanning several domains and topics. Each of the test and validation sets contains 1000 dialogues. As for automatic evaluation, we use *Inform Rate* and *Success Rate* to evaluate dialogue task completion. The former measures whether the system has provided a proper entity and the latter measures whether it has answered all the requested attributes (Budzianowski et al., 2018). Besides, *BLEU* (Papineni et al., 2002) is used to measure the fluency of generated responses. To measure the overall quality, we compute a combined score by (*Inform*+*Success*)×0.5+ *BLEU* (Mehri et al., 2019).

4.2 Implementation Details

Our model is trained on a 12 GB Nvidia GeForce RTX 2080 Ti with a batch size of 80. Our implementation[2] is based on PyTorch (Paszke et al., 2019). We pre-trained the BERT model based on the open-source library Transformers (Wolf et al., 2020). The dimension of word embeddings is 50 and the hidden size is 100. We use one-layer Bidirectional GRUs (Chung et al., 2014) as context encoders and three GRUs augmented with copy mechanism as decoders. The candidate actions are encoded by another Bidirectional GRU. We use Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.007. We use greedy search to decode system actions and beam search with a beam size of 5 to decode system responses. We use the ground truth belief states for a fair comparison with other baselines. We train our model for 60 epochs and select the best model on the validation set, and then evaluate it on the test set to get the final results.

4.3 Baselines

We compare our full model MAMD with several baselines on MultiWOZ 2.0: SC-LSTM (Wen et al., 2015), LaRL (Zhao et al., 2019), HDSA (Chen et al., 2019) , DAMD (Zhang et al., 2020), PARG (Gao et al., 2020), SimpleTOD (Hosseini-Asl et al., 2020), MarCo (Wang et al., 2020b), UBAR (Yang et al., 2020), HDNO (Wang et al., 2020a), LAVA (Lubis et al., 2020). Especially, SC-LSTM and HDSA treat system actions as one-hot vectors, and LaRL, HDNO, LAVA treats them as latent variables. Besides, HDSA uses BERT to predict system actions. DAMD, PARG, SimpleTOD, MarCo, and UBAR treat belief state, system actions as sequences and generate them along with system response. Besides, DAMD (aug) means DAMD using rule-based multi-action data augmentation to augment the system actions. Similar to HDSA, MarCo also uses BERT to predict system actions.

4.4 Overall Results

As shown in Table 1, our model significantly outperforms the baseline model DAMD in *Inform Rate*, *Success Rate* and especially *Combined Score*. Besides, our model achieves the best performance in *Combined Score* among all the baseline models. We also observe that models that generate system actions as a sequence generally have superior performance, implying that sequence is a better representation to model the inter-relationships among dialogue actions than one-hot vectors. Besides, our

[2]https://github.com/yunhaoli1995/MAMD-TOD

Model	DA	LM	Inform	Success	BLEU	Combined Score
SC-LSTM (Wen et al., 2015)	✗	✗	74.50	62.50	20.50	89.00
LaRL (Zhao et al., 2019)	✗	✗	82.80	79.20	12.80	94.10
SimpleTOD (Hosseini-Asl et al., 2020)	✗	✓	88.90	67.10	16.90	94.90
HDSA (Chen et al., 2019)	✗	✓	82.90	68.90	**23.60**	99.50
DAMD (Zhang et al., 2020)	✗	✗	89.50	75.80	18.30	100.90
DAMD (aug) (Zhang et al., 2020)	✓	✗	89.20	77.90	18.60	102.15
PARG (Gao et al., 2020)	✓	✗	91.10	78.90	18.80	103.80
MarCo (Wang et al., 2020b)	✗	✓	92.30	78.60	20.02	105.47
UBAR (Yang et al., 2020)	✗	✓	94.00	83.60	17.20	106.00
LAVA (Lubis et al., 2020)	✗	✗	**97.50**	**94.80**	12.10	108.25
HDNO (Wang et al., 2020a)	✗	✗	96.40	84.70	18.85	109.37
MAMD	✓	✗	95.70	88.90	18.90	**111.20**

Table 1: Overall results on the MultiWOZ 2.0 dataset. *DA* indicates whether to use data augmentation, and *LM* indicates whether to use pre-trained language models to predict system action.

Model	Inform	Success	BLEU	Score
SimpleTOD	85.10	73.50	16.22	95.52
HDSA	86.30	70.60	**22.36**	100.81
MarCo	92.50	77.80	19.54	104.69
UBAR	92.70	81.00	16.70	103.55
LAVA	**96.39**	83.57	14.02	104.00
HDNO	92.80	83.00	18.97	106.87
MAMD	94.20	**86.20**	18.80	**109.00**

Table 2: Overall results on the MultiWOZ 2.1 dataset.

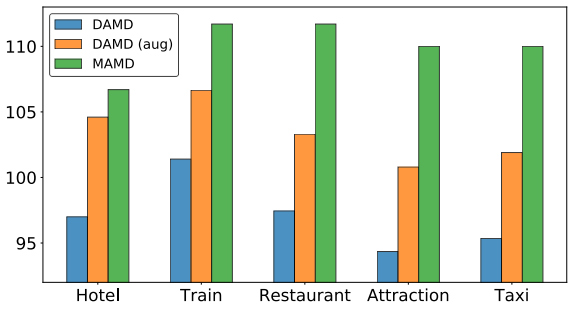

Figure 4: Results of our MAMD and DAMD in combined scores across different domains. If a dialogue involves more than one domain, it is counted into each.

model outperforms all the methods with data augmentation, which shows the effectiveness of our proposed retrieve-and-memorize framework.

We also evaluate our model on MultiWOZ 2.1 (Eric et al., 2020), an updated version of MultiWOZ 2.0. As shown in Table 11, the results are consistent with that on MultiWOZ 2.0 in Table 1.

4.5 Performance Across Different Domains

We report the performance of our model on different domains of MultiWOZ 2.0 and compare it with DAMD and DAMD (aug). The results are shown in Figure 4. From the bar chart, we can find that our model achieves the best performance across all domains. Besides, our model achieves significant performance improvements in *taxi* and *attraction* domains, which appear less frequently in the training data than other domains. Our MAMD narrows the performance gaps among different domains.

4.6 Ablation Study

In this section, we conduct experiments to study the contributions of the proposed context-aware retrieval module and memory-augmented module.

As shown in Table 3, the first group is the baseline directly trained on four types of augmented data, where it treats the augmented actions as equivalent to the golden ones. We observe that the performance drops significantly if the augmented actions are randomly selected, suggesting that the benefit of such data augmentation is strongly subject to the quality of the augmented data. Additionally, the model trained with CARM outperforms the Rule, which indicates the higher quality of our context-aware retrieved candidates and the effectiveness of the proposed CARM. What's more, removing the system actions prediction pre-training task in CARM causes a performance drop, which demonstrates the necessity to adjust the pre-trained model and obtain more task-related representations.

The second group in Table 3 shows the results of the model with the memory-augmented (MA) module trained as well as evaluated with various augmented data. First, with MA, our MAMD is much more robust to random noise, only slightly under-

Method	Score	Δ
Baseline	98.95	0
+ Random	91.65	-7.30
+ Rule	102.15	+3.20
+ CARM	106.65	+7.70
+ CARM w/o Pt	102.25	+3.30
+ Random + MA	98.10	-0.85
+ Rule + MA	106.75	+7.80
+ CARM + MA	108.70	+9.75
+ CARM + MA + RS	111.20	+12.25

Table 3: Results of ablation study. *Baseline* is MAMD without the memory-augmentation component. *Random* means randomly selected actions, *Rule* is the rule-based augmentation proposed by DAMD, *CARM* is the proposed context-aware retrieval module, and *w/o Pt* means without pre-training before retrieval. *MA* is the proposed memory-augmentation module, and *RS* is the proposed random sampling technique.

performing the baseline. This is because, during training, a model with MA can learn to ignore the noises in the memory and pay less attention to the memory during evaluation. Second, we see more performance gains with MA from both rule-based and context-aware retrieved candidates, which suggests a model with MA can utilize the candidate system actions more effectively. Last but not least, with the random sampling mechanism, the performance of our full model further improves.

4.7 Effect of Random Sampling

To further analyze the effect of random sampling, we adjust the random sampling probability during training from 0 (no random sampling and all candidates are from CARM) to 1 (all candidates are randomly sampled), and evaluate MAMD with retrieved candidates and randomly sampled candidates in the memory bank. As shown in Figure 5, the first thing to notice is that without random sampling, i.e., the random sampling probability p is set to 0, the performance of MAMD with random candidate system actions drops drastically to 66.40. This indicates MAMD trained with all decent-quality candidates has developed excessive dependence on the candidates and in a way treats them as ground truth actions, which is what we try to avoid by introducing random sampling. Once we introduced random sampling, the performance gap between MAMD evaluated with retrieved actions and random actions is significantly narrowed, which suggests MAMD is capable of telling the quality of the candidates in the memory bank.

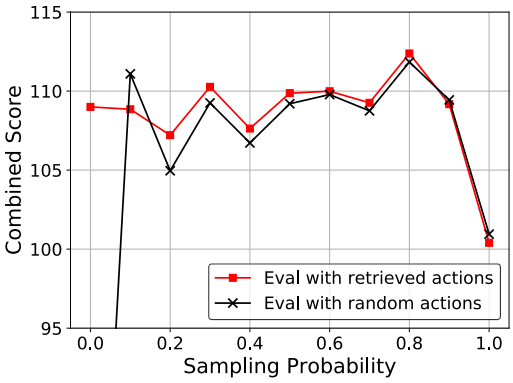

Figure 5: Results of our model trained with different random sampling probabilities and evaluated with different type of candidate actions on the development set.

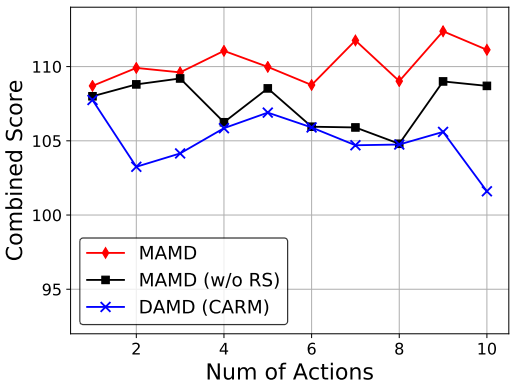

Figure 6: Combined score of three models trained with different numbers of candidate actions retrieved by CARM on the development set, where *MAMD (w/o RS)* means our model without random sampling and *DAMD (CARM)* means DAMD trained with augmented system actions retrieved by CARM.

4.8 Effect of the Number of Candidate Actions

To analyze the effect of the number of candidate actions on our proposed modules, we train three model variations with different numbers of candidate actions retrieved by CARM. As shown in Figure 6, we can see that both MAMD and MAMD (w/o RS) achieve their best performances with 9 candidate actions. Additionally, both our models consistently outperform DAMD, which suggests the effectiveness of the memory-augmented module. What's more, the performance of our full model increases more steadily as the number of candidate actions goes up, while without random sampling, the performance of our model is much more unstable across different numbers of candidate actions, which indicates that random sampling can bring in some desirable regularization.

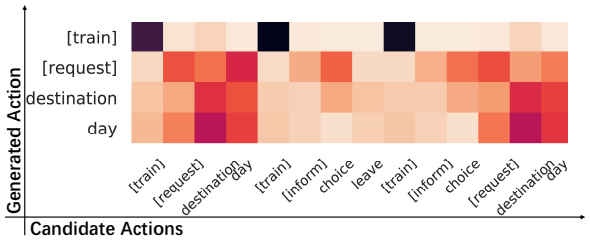

Figure 7: Visualization of the attention from generated system actions to candidate actions. The y-axis is generated system actions and the x-axis is candidate system actions. At each decoding step, the generated system actions selectively attend to the candidate actions. (Dialogue ID:MUL0473)

Context:	
Sys: I would recommend the cambridge museum of technology, would you like any information about that?	
User: Yes. What is the postcode and phone number?	
DAMD:	
[attraction] [recommend] postcode phone name type	
[attraction] [inform] phone postcode	
[attraction] [nooffer] type	
.....	
CARM:	
[attraction] [inform] phone postcode	
[attraction] [inform] postcode phone [general] [reqmore]	
Reference:	
[attraction] [inform] postcode phone [general] [reqmore]	

Table 4: Comparison of retrieved candidate system actions of DAMD and our CARM.

Context: ... User: Please book tickets and provide me with the total cost of tickets and confirmation number.
DAMD:
The [value_id] is **[value_price]**. The train id is [value_id]. Is there anything else I can help you with?
MAMD:
Booking was successful, the total fee is **[value_price]** payable at the station. Reference number is: **[value_reference]**. Is there anything else I can help you with?
Reference:
It has been booked! Your reference number is **[value_reference]**. The cost is **[value_price]**. Do you need anything else?

Table 5: An example of response generation of DAMD and MAMD.

5 Visualization and Case Study

An illustrative example is shown in Figure 7, the current user utterance is "*I need a train departing cambridge arriving by 20:30*". The action decoder successfully attends to appropriate actions and ignores the noisy ones like "*[train] [inform] leave*", as the leaving time has not provided by the user.

Table 4 shows an example of candidate system actions that CARM appropriately retrieved but DAMD failed. The user asks the system to provide the postcode and phone number of the attraction, while DAMD returns "*[attraction][nooffer][type]*".

We also present an example of response generation in Table 5, where the user asks for the price and reference number. DAMD manages to provide the postcode but fails to provide the reference number, while our MAMD model successfully provides both the postcode and the reference number.

MAMD vs. DAMD	Win%	Tie%	Lose%
Completion	19.25%	66.54%	14.21%
Readability	3.13%	93.08%	3.79%

MAMD vs. Reference	Win%	Tie%	Lose%
Completion	14.51%	56.11%	29.38%
Readability	2.85%	92.03%	5.11%

Table 6: Results of human evaluation on response quality. *Reference* means ground truth response. *Win*, *Tie* and *Lose* respectively indicate the proportions that our model wins over, ties with or loses to its counterpart.

6 Human Evaluation

Finally, we conduct a human study to evaluate our model from the human perspective. We randomly select 30 dialogue sessions (211 dialog turns in total) from the test dataset and have 5 postgraduates as judges to compare two groups of systems: MAMD vs. DAMD and MAMD vs. Reference, in terms of *Readability* and *Completion* (Wang et al., 2020b). *Completion* measures whether a response has correctly answered a user query, including relevance and informativeness. *Readability* measures the fluency and consistency of the response.

We report the human evaluation results in Table 6, from which we can observe that our model outperforms DAMD and beats or ties with Reference nearly 70% of the time in terms of *Completion*. In *Readability*, our model ties more than 92% with DAMD as well as Reference. This may suggest the language of responses lacks diversity and is easy to learn. Overall, our model is superior to DAMD in human evaluation, which demonstrates its competence in a more holistic evaluation other than automatic metrics.

7 Conclusion

In this paper, we proposed a retrieve-and-memorize framework to deal with the unbalanced distribution of system actions in task-oriented dialogue systems. Our framework includes a neural retrieval module that can retrieve multiple candidate system actions given a dialogue context, and a memory-augmented multi-decoder network that can generate system actions conditioned on multiple candidate system actions. Extensive experiments were conducted on a large-scale multi-domain task dialogue dataset and the results demonstrate the effectiveness of our framework. In essence, the whole framework, including its random sampling strategy, can be viewed as an attempt to prevent the systems from overfitting skewed dialogue datasets with an unbalanced distribution of system actions.

Acknowledgments

The paper was supported by the National Natural Science Foundation of China (No.61906217) and the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (No.2017ZT07X355).

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. MultiWOZ-A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3696–3709.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Silin Gao, Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Paraphrase augmented task-oriented dialog generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 639–649, Online. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1631–1640.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414.

Weixin Liang, Youzhi Tian, Chengcai Chen, and Zhou Yu. 2020. MOSS: End-to-end dialog system framework with modular supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8327–8335.

Nurul Lubis, Christian Geishauser, Michael Heck, Hsien-chin Lin, Marco Moresi, Carel van Niekerk, and Milica Gasic. 2020. LAVA: Latent action spaces via variational auto-encoding for dialogue policy optimization. In *Proceedings of the 28th International Conference on Computational Linguistics*,

pages 465–479, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. 2019. Structured fusion networks for dialog. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 165–177.

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 of the Association for Computational Linguistics*, pages 311–318.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020a. Modelling hierarchical structure between dialogue policy and natural language generator with option framework for task-oriented dialogue system. *arXiv preprint arXiv:2006.06814*.

Kai Wang, Junfeng Tian, Rui Wang, Xiaojun Quan, and Jianxing Yu. 2020b. Multi-domain dialogue acts and response co-generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7125–7134, Online. Association for Computational Linguistics.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: paraphrase generation with semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7176–7183.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2020. UBAR: Towards fully end-to-end task-oriented dialog systems with GPT-2. *arXiv preprint arXiv:2012.03539*.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, pages 1208–1218.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1458–1467, Melbourne, Australia. Association for Computational Linguistics.

A More Details of MAMD

A.1 Attention Function

In MAMD, we use an attention function *Attn* to attend to three groups of hidden states. In this study, $Attn(h, H_a, H_b, H_c)$ is defined as:

$$\begin{aligned} h_a &= \text{CatAttn}(h, H_a), \\ h_b &= \text{CatAttn}(h, H_b), \\ h_c &= \text{CatAttn}(h, H_c), \\ \text{Attn}(h, H_a, H_b, H_c) &= [h_a \oplus h_b \oplus h_c], \end{aligned} \tag{12}$$

where $\oplus$ is the concatenation operator and CatAttn is a simple concat-attention defined as:

$$\begin{aligned} a_i &= \tanh(W[h \oplus H_i]), \\ \alpha_i &= \text{Softmax}(a), \\ \text{CatAttn}(h, H) &= \sum\nolimits_{i=1}^{n} \alpha_i H_i, \end{aligned} \tag{13}$$

where W represents learnable parameters, H is the sequence of encoded hidden states,[3] and n is the number of hidden states in H.

A.2 Decoder with Copy Mechanism

The decoder used to generate the belief state, system action and response is a one-layer GRU augmented with copy mechanism. Each step of the

[3]For example, the encoded hidden states of user utterance.

generation in $\text{Dec}(c_t, h_{t-1}, H)$ is defined as follows:

$$
\begin{aligned}
h_t &= \text{GRU}(c_t, h_{t-1}), \\
p_{vocab} &= \text{Softmax}(W_v h_t), \\
s_i &= h_t^{\top} \tanh(W_c H i), \\
p_{copy} &= \text{Softmax}(s), \\
p_{final}(w) &= p_{vocab}(w) + \sum_{i:X(i)=w} p^i_{copy}, \\
\text{Dec}(c_t, h_{t-1}, H) &= p_{final}, h_t,
\end{aligned}
\tag{14}
$$

where W_v and W_c are learnable weights, and X is the corresponding context of H.

B More implementation Details

B.1 Hyperparameters

In this section, we report the hyperparameter setting in our model. For MAMD, we adopt the default hyperparameters in DAMD, as shown in Table 7. As for the learning rate, the number of candidate actions, and the random sampling probability, we apply grid search to find the best combination on the development set. It takes about 10 hours to train our model on a single 12 GB Nvidia GeForce RTX 2080 Ti. As for CARM's pre-training task, the hyperparameter setting is shown in Table 8.

Parameter	Values
batch size	80
learning rate	7e-3
embedding size	50
hidden size	100
dimension of db search result	6
encoder layers	1
decoder layers	1
epoch	60
candidate actions	9
random sampling probability	0.8
beam size	5
random seed	777

Table 7: Hyperparameter setting of MAMD.

B.2 Delexicalization Strategy

For delexicalization, we follow DAMD's domain-adaptive delexicalization strategy. Specially, we use tokens such as [value_name] to represent the same slot name. In this case, the placeholders [hotel_name] and [restaurant_name] will be converted to [value_name]. During the evaluation, we induce the domain of a placeholder from the transition between two adjacent belief states and the generated system actions of the current dialog turn.[4]

Parameter	Values
batch size	6
learning rate	5e-5
epoch	20
random seed	42
max sequence length	400
warmup proportion	0.1

Table 8: Hyperparameter setting of CARM.

B.3 Post-Processing of Candidate Action Retrieval

As for candidate action retrieval, we retrieve 50 candidate actions for each sample. Then, we apply post-processing to clean the candidate actions:

- Duplicated actions are merged. For example, the system actions "[attraction] [inform] postcode phone [general] [reqmore]" and "[attraction] [inform] postcode phone [general] [reqmore]" will be combined into "[attraction] [inform] postcode phone [general] [reqmore]".
- Null system actions are removed.
- System actions with different database query results are filtered out.
- System actions that conflict with current belief are filtered, e.g., requesting a slot that is already included in belief states.

C Dataset Details

We provide more information about the MultiWOZ 2.0 dataset. The training set contains 8438 dialogs, 115,424 turns, and 1,520,970 tokens. The average number of turns per dialog is 13.68, and the average number of tokens per turn is 13.18. The number of slots and values are 25 and 4510, respectively. The ontology is shown in Table 9. We also count the numbers of system actions across different domains. As shown in Figure 8, the numbers of system actions in *attraction* and *taxi* are smaller than the other domains, showing the unbalanced distribution of system actions at the domain level.

[4]For more details, please refer to the source code.

act type	inform* / request* / nooffer[1234] / recommend[123] / select[1234] / offerbook[124] / offerbooked[124] / nobook[12] / bye* / greet* / reqmore* / welcome*
slot	car[5] / address[12367] / postcode[12367] / phone[123567] / internet[2] / parking[2] / type[23] / pricerange[12] / food[1] / stars[2] / area[123] / reference[1234] / time[14] / leave[45] / price[45] / arrive[45] / id[4] / stay[2] / day[124] / leave[45] / people[123] / name[123] / destination[45] / departure[45] / department[6]

Table 9: Ontology for all domains. The upper script indicates which domains it belongs to (∗: universal, 1: restaurant, 2: hotel, 3: attraction, 4: train, 5: taxi, 6: hospital, 7: policy).

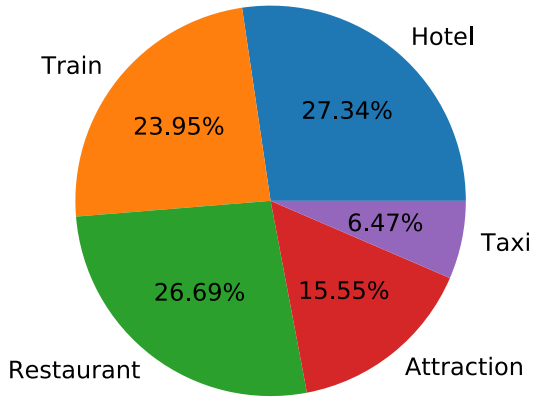

Figure 8: Statistics of system actions across different domains of MultiWOZ 2.0.

Dateset	Inform	Success	BLEU	Score
Development	96.60	90.70	18.70	112.35
Test	95.70	88.90	18.90	111.20

Table 10: Overall results on the MultiWOZ 2.0 dataset.

Dateset	Inform	Success	BLEU	Score
Development	94.90	87.70	18.60	109.90
Test	94.20	86.20	18.80	109.00

Table 11: Overall results on the MultiWOZ 2.1 dataset.

D More Analyses and Discussions

D.1 Results on Development and Test Sets

We report the results of MAMD on the development and test sets of MultiWOZ 2.0 and MultiWOZ 2.1. As shown in Table 10 and Table 11, the results on the development set are generally consistent with that on the test set on both benchmarks.

D.2 Distribution of Generated System Actions

To further analyze the influence of our model on the generation of system actions, we count the appearance of generated actions. Recall that each dimension of the actions stands for either *domain*, *function* or *slot*, where *domain* defines the domain involved in the conversation, and *function* defines the behavior of system such as informing the user or request certain information. Here we only count the first two dimensions of the actions because the third dimension appears to be less important.

As shown in Figure 9, the distribution of system actions generated by DAMD is proportional to the original distribution in the dataset, and DAMD tends to generate fewer actions than the original distribution. After applying their rule-based multi-action data augmentation, DAMD (aug) can generate more diverse system actions compared with DAMD. Compared with DAMD (aug), MAMD generates more actions. More importantly, MAMD generates more important actions such as "attraction-inform" and "taxi-inform" which are more relevant to task completion, while DAMD (aug) tends to generate less useful actions such as "general-require" and "general-greet". This phenomenon indicates that the memory-augmented mechanism provides some guidance to our model during system action learning. To sum up, our proposed model can generate more diverse and valuable actions, which demonstrates the effectiveness of our proposed memory-augmented mechanism.

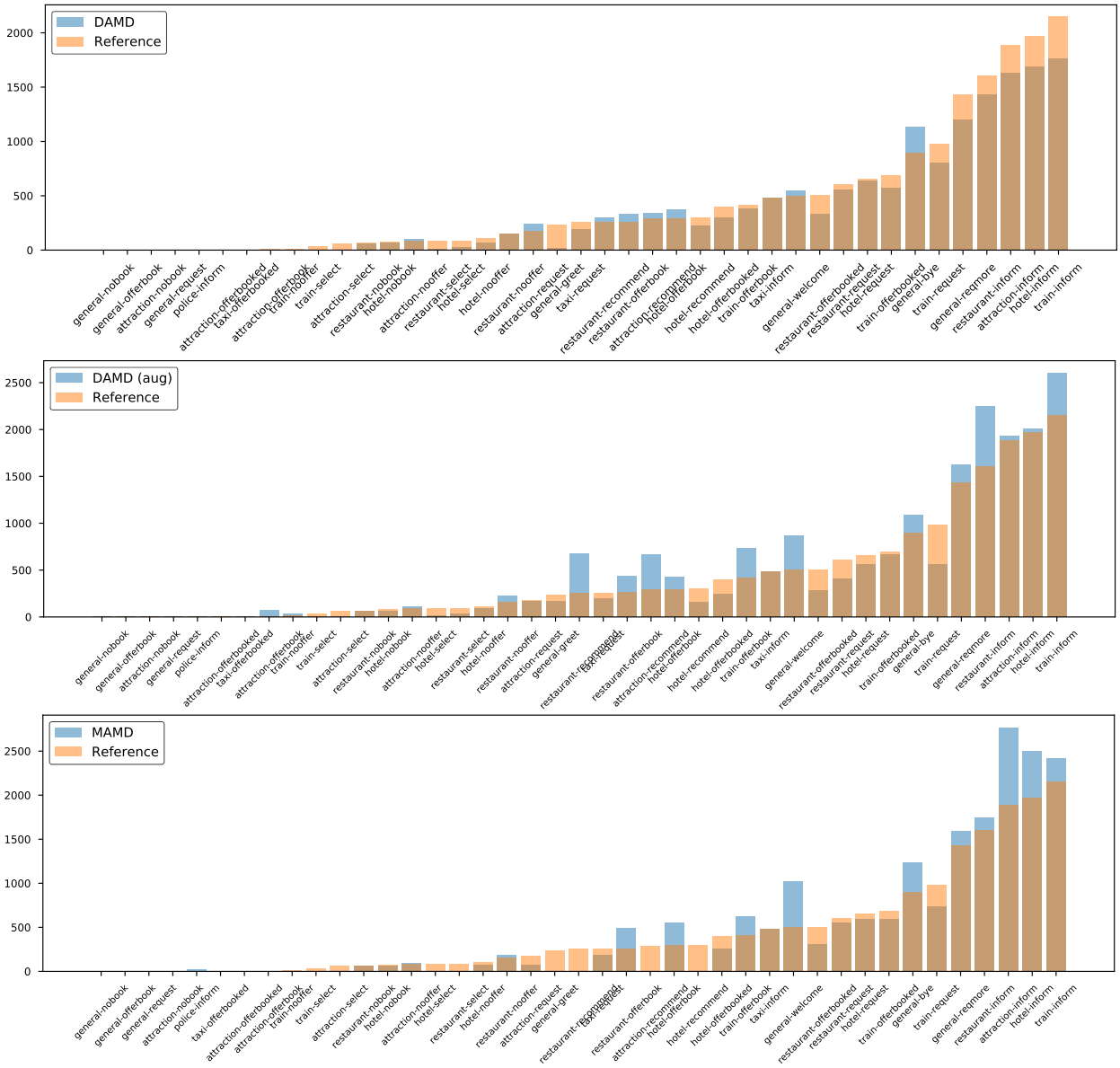

Figure 9: Statistics of generated system actions by DAMD, DAMD (aug) and MAMD, and comparison with reference actions.