Improve Zero-shot Performance on Unseen Dialog Domains with Retrieval-Augmented Task-Oriented Systems

Anonymous SIGDIAL submission

Abstract

End-to-end task-oriented dialog (TOD) systems have achieved promising performance by leveraging sophisticated natural language understanding and natural language generation capabilities of pre-trained models. This work enables the TOD systems with more flexibility through a simple cache. The cache provides the flexibility to dynamically update the TOD systems and handle unseen dialog domains. Towards this end, we first fine-tune a retrieval module to effectively retrieve the most relevant information entries from the cache. We then train end-to-end TOD models that can refer to and ground on both dialog history and retrieved information during TOD generation. The introduced cache is straightforward to construct, and the backbone models of TOD systems are compatible with existing pre-trained generative models. Extensive experiments demonstrate the superior performance of our framework, with a notable improvement in unseen joint goal accuracy by 6.7% compared to strong baselines.

1 Introduction

Task-oriented dialog (TOD) systems play an important role in various applications, such as restaurants booking, alarm setting, and recommendations (Gao et al., 2018; Xie et al., 2022). These systems can be broadly categorized into two groups: pipeline-based dialog systems and end-to-end dialog systems. Pipeline-based dialog systems consist of four separate modules, namely a natural language understanding (NLU) module for detecting user intents, a dialog state tracking (DST) module to track user belief states across dialog turns, a dialog management (DM) module for system actions based on dialog states, and a natural language generation (NLG) module for generating natural-language responses. However, the pipeline-based approach is label-intensive, prone to error propagation, and challenging to scale (Hosseini-Asl et al., 2020; Peng et al., 2021; Feng et al., 2023).

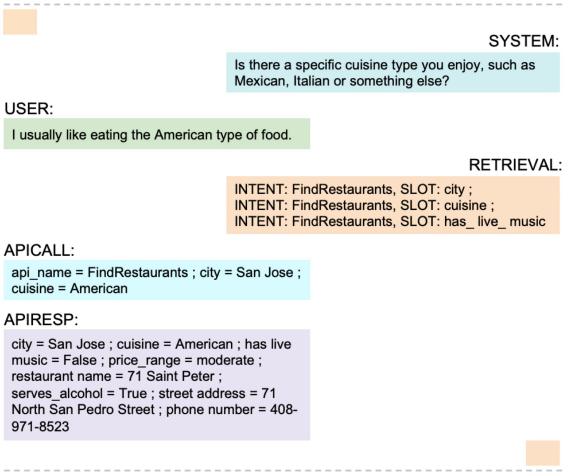

Figure 1: An example of the auto-regressive TOD with retrieved slot information from cache. The APICALL generation process is shown, with N set to 3 for the retrieval module.

Recently, various approaches have been proposed to utilize sequence-to-sequence models for generating dialog states and responses in an end-to-end manner (Ham et al., 2020; Lin et al., 2020; Yang et al., 2021; Gao et al., 2021; Chen et al., 2021; Peng et al., 2021; Liu et al., 2021; He et al., 2022b; Feng et al., 2023). Compared with the pipeline-based systems, these approaches demonstrate effectiveness on public datasets with fewer direct annotations required, such as user intents and dialog acts. Additionally, they leverage the capabilities of large-scale pre-trained language models, such as GPT-2 (Radford et al., 2019), T5 (Raffel et al., 2019), and BART (Lewis et al., 2020a), for improved performance in NLU and NLG tasks. However, these approaches are limited in their ability to dynamically handle existing, unseen, or emerging intents and slots, particularly in the context of unseen dialog domains (Hosseini-Asl et al., 2020; Peng et al., 2021). This limitation hinders their zero-shot performance on unseen domains,

where the model has not been explicitly trained or exposed to the specific dialog domain.

In parallel, research on open-domain question answering and dialog systems has explored the use of retrieval-augmented models. These models retrieve relevant information from a passage, database, APIs, etc., and incorporate it into the generation process, improving answer quality or dialog responses (Karpukhin et al., 2020; Izacard and Grave, 2021; Dinan et al., 2018; Lewis et al., 2020b; Shuster et al., 2021). Inspired by these ideas, we combine both worlds and propose an end-to-end TOD framework with a retrieval system that addresses the challenge of handling zero-shot unseen dialogs.

Our approach involves training the end-to-end TOD models with a cache that contains accessible domains, intents, slots and APIs. The cache can be constructed based on the schema or database, or by extracting information from accessible dialogues when the schema or database is not fully accessible. The cache serves as a reference point, allowing the models to ground their responses in the retrieved information. By incorporating a retrieval module and leveraging this cache of knowledge, our system enhances the flexibility and adaptability to handle both existing and unseen intents and slots, and enables robust performance even in novel dialog domains where the model has not been explicitly trained. Figure 1 shows an illustrative example of our approach, demonstrating how the RETRIEVAL module retrieves relevant information, such as slots in this case, from the cache to enrich the system's understanding and generate more accurate responses. The APICALL represents the dialog states from the system side, and APIRESP returns information from external API interactions between the system and system databases.

To build an accurate end-to-end TOD system with the benefits of a simple cache, we fine-tune a retrieval module to effectively retrieve the most relevant and informative information from the cache, using a Top-N retrieval strategy. Then we integrate the retrieval module into the generative model to facilitate end-to-end TOD generation. We evaluate our approach on the publicly available Google Schema-Guided Dialogue dataset (SGD) (Rastogi et al., 2020b), which includes a significant number of unseen dialog domains in the development and test sets.

The contributions of this paper are as follows: (1) We design a simple yet effective end-to-end TOD framework with a cache that enables dynamic handling of intents and slots. The framework is compatible with existing pre-trained generative models, and enhances the system's robustness in handling unseen dialog domains. (2) Experimental results demonstrate the superior performance of our approach compared to strong baselines. It achieves 6.7% improvement in zero-shot joint goal accuracy, demonstrating the effectiveness in handling unseen dialog domains. (3) To advance future research in accurate end-to-end TOD systems, we conduct comprehensive ablation studies and analyses to provide insights into the impact of different components and design choices within our framework.

2 Related Work

End-to-End TOD Systems End-to-end TOD models have shown promising performance on public dataset (Ham et al., 2020; Lin et al., 2020; Yang et al., 2021; Gao et al., 2021; Chen et al., 2021; Peng et al., 2021; Liu et al., 2021; He et al., 2022a,b; Feng et al., 2023; Bang et al., 2023). These approaches typically follow common patterns: (1) Rely on powerful pre-trained seq2seq models. (2) Use language modeling objectives to generate NLU and NLG outputs, sometimes augmented with auxiliary multi-task goals like DST loss. (3) Either fine-tune models directly on the target dataset or conduct pre-training on multiple TOD dialogue datasets. (4) Employ data augmentation techniques such as back-translation and entity replacement due to the challenges in collecting large-scale TOD corpora. For example, Hosseini-Asl et al. (2020) propose a simple language model for TOD based on DistilGPT2. The model generates user belief states and system responses in an auto-regressive way. Peng et al. (2021) introduce two auxiliary tasks for belief state prediction and grounded response generation and pre-train language models first on multitple TOD dataset. Gao et al. (2021) enables the belief state to interact with both structured and unstructured knowledge. Feng et al. (2023) designs a reward-function learning objective to guide the model's generation. While these methods have demonstrated effectiveness on public datasets, they have limitations in handling unseen dialog domains.

Retrieval-Augmented Models Retrieval augmented approaches have been widely used in

open-domain question answering. For instance, Karpukhin et al. (2020) propose a BERT-based (Devlin et al., 2019) dual-encoder framework to retrieve passages from Wikipedia, which is further incorporated into open-domain conversations to reduce hallucination and enrich engagement with users (Shuster et al., 2021; Komeili et al., 2021). These models retrieve information related to the query from a knowledge base of sentences and ground the generation response on this information (Dinan et al., 2018; Lewis et al., 2020b). Inspired by these works, we explore the integration of retrieval modules into end-to-end TOD systems, leveraging the retrieval-augmented approach to enhance the system's performance in handling unseen dialog domains.

3 TOD Systems with a Simple Cache

We present an end-to-end transformer-based framework with a simple cache that is compatible with multiple generative models, including BART, T5, GPT2, etc. Our framework enables dynamic handling of intents, slots, and APIs while maintaining flexibility in choosing the backbone model.

Generally, our framework consists of two parts: a retrieval model for retrieving the most relevant and informative information from the cache, and an end-to-end TOD model that generates APICALLs and system responses based on the dialog history and the retrieved information. The retrieval model functions by retrieving intents, slots, APIs, and other relevant information from the cache.

Figure 2 illustrates one simple variant of our framework, which is an encoder-decoder architecture. In this variant, the retrieved information such as slots are stacked together. We also introduce another variant in Sec. 4.2, where each retrieved information is concatenated with the dialog history and then all the information are concatenated together before being sent to the decoder.

3.1 Construction of Cache

In this section, we describe the construction of a simple cache that provides necessary information for the model's referencing and grounding procedure. The cache consists of intents, slots, and APIs extracted from the schema and database. In cases where the schema or database is not fully accessible, we extract information from accessible dialogues. During training, it is important to note that the cache exclusively contains information relevant to the training dialogues and does not incorporate any unseen information of dialogues in the test set.

Since there are different ways to construct a cache, we design various templates to formalize the retrieved information. Table 1 presents several templates that we utilize. One example we is the "API-description" template, where an API includes all the intents and relevant slots mentioned throughout the whole dialogue. Although this template may contain redundant information as some intents and slots may not be mentioned initially, it allows us to evaluate the model's ability to disregard irrelevant details.

In addition to the listed templates, we explore several other templates with special tokens such as "*[INTENT] intent name [SLOT] slot name*", as well as different orderings of intents and slots, such as "*intent name, intent description, slot name, slot description*" and "*intent name, slot name, intent description, slot description*". We conduct an in-depth analysis of the effects of different cache templates in the experimental section.

3.2 Retrieval Module

After constructing the cache, we fine-tune the retrieval model to effectively retrieve the most relevant and informative information for the dialogue context. Given a dialog history c, the TOD system utilizes a retrieval module to retrieve Top-N most relevant information $s_1, \ldots, s_N$ from the cache. Firstly, based on the user dialog history, the system triggers the retrieval module to generate an APICALL, which includes relevant mentioned intents, slots and values. Subsequently, the system continues to utilize the retrieval module to generate a system response based on all previous information.

To ensure accurate retrieval from the cache, we fine-tune a dense passage retriever (DPR) model (Karpukhin et al., 2020), which is a BERT-based dual-encoder framework optimized via contrastive learning. Specifically, we obtain the hidden representation $\mathbf{h}_c$ for the dialog history using an encoder model, *e.g.*, $\mathbf{h}_c = \text{BERT}_c(c)$. Similarly, we use another BERT encoder to obtain the feature representation $\mathbf{h}_s$ for each retrieved information entry from the cache, *i.e.*, $\mathbf{h}_s = \text{BERT}_s(s)$. The similarity between the dialog history and the retrieved information entry is: $sim(c, s) = \mathbf{h}_c^T \odot \mathbf{h}_s$.

For each dialog history, there are n relevant (positive) entries and m irrelevant (negative) entries, where n and m may vary as each dialog history

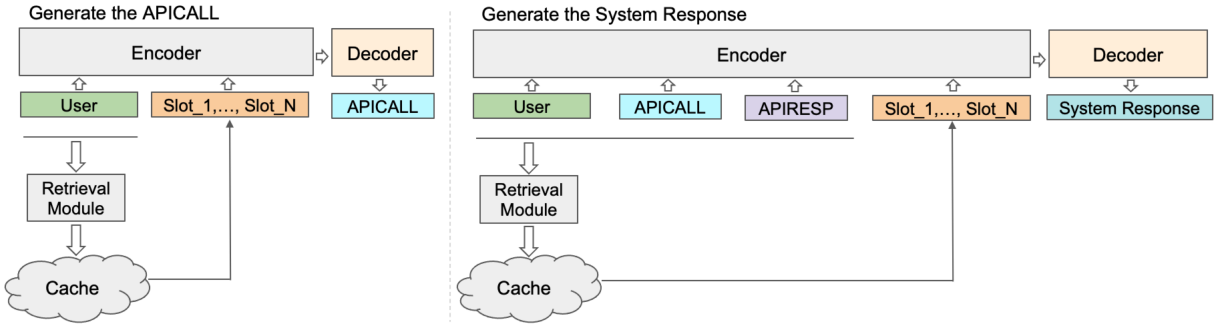

Figure 2: Illustration of the end-to-end framework with a simple cache. The left figure shows the generation of an APICALL, with the retrieval module extracting most relevant information such as slots from the cache. The retrieved information, combined with dialog history, is used by the decoder to generate the APICALL. The right figure depicts the continuation of the dialogue, generating the system response. The system retrieves additional information from the cache, and incorporate all previous information to generates a system response. The decoupling of APICALL generation and system response generation aims to provide a clear representation of the framework's components and their interactions in an end-to-end setting.

Cache Templates	Examples
INTENT: intent name, SLOT: slot name	INTENT: findrestaurants, SLOT: city
intent name, slot name, service description, intent description, slot description	findrestaurants, city, a leading provider for restaurant search and reservations, find a restaurant of a particular cuisine in a city, city in which the restaurant is located
API-descriptions	api_name = FindRestaurants; optArg = has_live_music, price_range, serves_alcohol; reqArg = city, cuisine

Table 1: Several typical templates of the simple cache construction, where each template represents one type of cache. Some other templates can be found in Table 4.

would contain different active intents and slots. Our objective is to learn a function that minimizes the distance between pairs of relevant dialog histories and information entries than the irrelevant pairs. The corresponding loss function for a specific pair is as follows:

$$\mathcal{L}_{\text{api}}(c, s_1^+, s_1^-, \ldots, s_m^-) = -\log \frac{\exp(\text{sim}(\mathbf{h}_c, \mathbf{h}_{s_1^+}))}{\sum_{j=1}^{m} \exp(\text{sim}(\mathbf{h}_c, \mathbf{h}_{s_j^-}))} . \quad (1)$$

Once the retrieval module is fine-tuned, it is integrated into the end-to-end TOD generative model. The parameters of the retrieval module remain fixed during training of the generative model.

Negative Sampling In the training process, we employ negative sampling to include retrieved information entries that are irrelevant to the dialog history. We utilize both natural and hard negative pairs to enhance the robustness and performance of the retrieval module.

For natural negative pairs, we consider pairs such as "*irrelevant intent, irrelevant slots*" as counterparts to the positive pairs of "*relevant intent, relevant slots*". Additionally, we construct hard negative pairs that pose a more challenge to the retrieval module. These hard negative pairs include combinations such as "*relevant intent, irrelevant slots from the same relevant intent*" and "*irrelevant intents that are semantically similar to the relevant intent, along with relevant slots from the relevant intent*". By incorporating these hard negative pairs, we encourage the retrieval module to learn to differentiate between relevant and irrelevant information effectively.

3.3 End-to-End TOD Systems

Our end-to-end TOD framework generates the APICALL and system response in an auto-regressive manner. Figure 1 provides an example of this process. The APICALL represents the dialog states from the system side, and same with previous work (Hosseini-Asl et al., 2020; Peng et al., 2021), it is an intermediate step of the system response generation, and they share the same model framework to generate tokens autoregressively.

For each dialogue turn, the TOD framework triggers the retrieval module twice. The system first retrieves the Top-N information entries from the constructed cache, *i.e.*,

$$\text{Top-}N \text{ info} = \text{Retrieval}(c) . \quad (2)$$

Then it generates an APICALL using the retrieved

information, *i.e.*,

$$\text{APICALL} = \text{TOD}(c, \text{Top-}N \text{ info}) . \quad (3)$$

After that the TOD framework retrieves another set of Top-N information entries from the cache, considering the generated APICALL, *i.e.*,

$$\text{Top-}N \text{ info} = \text{Retrieval}(c, \text{APICALL}, \text{APIRESP}) , \quad (4)$$

where APIRESP is automatically obtained from corresponding API, without the need for prediction.

Finally, the system generates a system response using the following inputs:

$$\text{Response} = \text{TOD}(c, \text{APICALL}, \text{APIRESP}, \text{Top-}N \text{ slots}) . \quad (5)$$

4 Experimental Settings

4.1 Dataset

A substantial number of end-to-end TOD works (Hosseini-Asl et al., 2020; Peng et al., 2021; Lin et al., 2020; Yang et al., 2021; Su et al., 2021; He et al., 2022b; Feng et al., 2023) commonly employ the MultiWOZ datasets (Budzianowski et al., 2018; Zang et al., 2020). However, these studies primarily focus on full-shot and few-shot learning, with less emphasis on zero-shot evaluation. This scope for zero-shot evaluation appears somewhat constrained given that MultiWOZ only has five domains and approximately 35 slots, all of them are presented in the training set. In contrast, our work aims to assess the system across large-scale, unseen dialog domains. We utilize the Google Schema-Guided Dialogue (SGD) dataset (Rastogi et al., 2020c). SGD provides a more expansive dialog landscape, with over 16k multi-domain conversations across more than 16 domains and 200 slots. Importantly, half of these domains and slots do not appear in the development and test sets. [1]. Table 2 summarizes the statistics of SGD.

4.2 Models

In term of baselines, we adopt (Lin et al., 2020; Chen et al., 2021) and implement their model "MinTL (BART-Large)". We also implement "T5DST" from (Lee et al., 2022), which achieves SOTA performance on MultiWOZ 2.2 (Zang et al., 2020). Since our end-to-end TOD framework is compatible with existing pre-trained generative models, we experiment with BART, GPT2 and T5.

[1] https://github.com/google-research-datasets/dstc8-schema-guided-dialogue

	Dialogues	Domains	Services	ZS Domains	ZS Services
Train	16142	16	26	-	-
Dev.	2482	16	17	1	8
Test	4201	18	21	3	11

Table 2: Data Statistics of SGD. ZS: Zero-Shot.

Interestingly, we found that BART-Large (406M) perform comparably with T5-Large (770M), despite having fewer parameters. Moreover, it outperformed SOTA models developed by teams in DSTC8 (Rastogi et al., 2020a), where the majority of models are BERT-based classification models. Thus, we select BART-Large as our primary backbone model.

Inspired by previous model designs in open-domain question answering (Lewis et al., 2020b; Izacard and Grave, 2021), we design two variants for end-to-end TOD systems. The first, named Fusion-in-Decoder TOD ("FiD-TOD"), is illustrated in Figure 2, In this model, the retrieved information such as slots, are stacked together. Notably, when the retrieval model is not incorporated, "FiD-TOD" becomes identical to "MinTL (BART-Large)". The second variant "FiD-TOD NoStack", is depicted in Figure 3 and is used as ablation study. In this model, the retrieved information is not directly stacked, and instead, the dialog history is concatenated with each retrieved information entry and then sent to the shared encoder.

Regarding the generative model, we truncate the tokens of dialog history to 256, and retrieve Top-5 most relevant information entries from the cache, unless otherwise specified. For DPR fine-tuning, we align one hard negative pair to each positive pair. We employ the preset hyperparameters from the ParLAI code (Miller et al., 2017), such as setting the learning rate to 5e-5, batch size to 32, etc. We selected the best model based on its performance on the development set.

The retrieve model is fine-tuned up to 3 epochs based on open-sourced DPR (Karpukhin et al., 2020), and the generative model is fine-tuned up to 4 epochs with an overall batch size of 64 on 8 Nvidia Tesla V100 GPUs. To facilitate reproducibility, all experiments are based on public code from the ParLAI platform[2].

4.3 Evaluation Metrics

We evaluate the end-en-end TOD framework using the following metrics: (1) Top-N accuracy: It

[2] https://github.com/facebookresearch/ParlAI

	PPL	Overall JGA	Unseen JGA	Token EM	BLEU-4
MinT (BART-Large) (Chen et al., 2021)	2.385	0.812	0.364	0.497	**0.179**
T5DST (Lee et al., 2022)	2.419	0.810	0.361	0.491	0.170
FiD-TOD	**2.133**	**0.829**	**0.431**	**0.501**	**0.179**

Table 3: Testing results on the SGD dataset.

Cache Templates	Top-1	Top-2	Top-3	Top-4	Top-5
INTENT: intent name, SLOT: slot name	0.833	0.882	0.914	0.945	0.960
INTENT: intent name, service description, intent description, SLOT: slot name, slot description	0.887	0.922	0.952	0.976	0.980
intent name, slot name, intent description, slot description	0.835	0.906	0.928	0.946	0.955
intent name, slot name, service description, intent description, slot description	0.913	0.943	0.965	0.977	0.981
API-descriptions	0.844	0.927	0.956	0.962	0.967

Table 4: Top-5 retrieval accuracy on the test set of SGD.

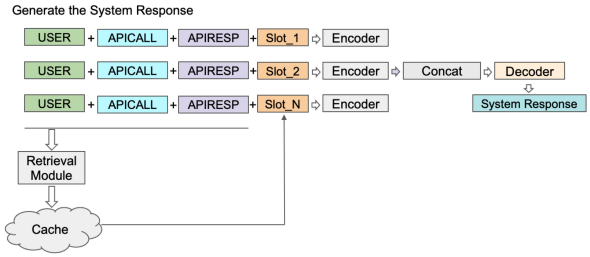

Figure 3: Illustration of the end-to-end "FiD-TOD NoStack" framework.

evaluates the retrieval module through checking whether the ground-truth slot appears in the Top-N predicted candidates (Karpukhin et al., 2020). (2) Joint Goal Accuracy (Overall JGA): It evaluates whether the predicted APICALL on both seen and unseen domains is correct or not, specifically. JGA is 1 if the model correctly predicts all intent, slots and corresponding values in the APICALL. Otherwise, JGA is 0. (3) Unseen JGA: It evaluates whether overall JGA is correct if the model calls the API on unseen domains. In SGD, most dialogue turns would not trigger an API retrieval, resulting in empty APICALLs, and identifying Empty JGA is relatively easy (Chen et al., 2021). Therefore, we focus on non-empty APICALL turns and treat Unseen JGA as the most crucial metric on unseen domains. (4) Token EM: It evaluates the utterance-level token accuracy. Roughly corresponds to perfection under greedy search (generative only). (5) Perplexity (PPL): it measure the generative model's ability to predict individual tokens. (6) BLEU-4: It measures the BLEU score (Papineni et al., 2002) between the predicted system response and the reference response. [3]

5 Experimental Results

5.1 End-to-End TOD Performance

Table 3 shows the overall performance on the test set. "FiD-TOD" outperforms baselines across most metrics. Specifically, it improves the essential NLU metric, *i.e.*, Unseen JGA, by 6.7%. This demonstrates the model's enhanced capability in handling zero-shot unseen dialog domains. As "MinT (BART-Large)" corresponds to the "FiD-TOD" without the retrieval model from the cache, this comparison highlights the significant benefits that our design brings to the handling of unseen dialogs. Additionally, the other metrics related to NLG are also slightly improved.

5.2 Retrieval Performance

We hope the model can generalize well as there could be many new intents and slots in real world.

General As shown in Table 4, our model shows effective Top-5 retrieval accuracy on the test set, keeping in mind that more than half of the services and slots are unseen in this set. The model shows good Top-1 accuracy and above 96% Top-5 accuracy, demonstrating strong abilities for handling both seen and unseen intents and slots. Compared to only using names, adding related service and intent descriptions improves the Top-1 accuracy

[3] Most metrics can be found in https://parl.ai/docs/tutorial_metrics.html.

	PPL	Overall JGA	Unseen JGA	Token EM	BLEU-4
MinT (BART-Large) (Chen et al., 2021)	1.700	0.876	0.586	0.538	**0.221**
INTENT: intent name, SLOT: slot name	1.688	0.889	0.633	0.538	0.212
intent name, slot name, intent description, slot description	1.679	0.895	**0.661**	0.541	0.215
intent name, slot name, service description, intent description, slot description	1.679	0.894	0.649	0.541	0.212
INTENT: intent name, service description, intent description, SLOT: slot name, slot description	**1.676**	**0.897**	0.660	**0.545**	0.217

Table 5: Performance of FiD-TOD on the development set with variations of cache templates.

	PPL	Overall JGA	Unseen JGA	Token EM	BLEU-4
MinT (BART-Large) (Chen et al., 2021)	1.700	0.876	0.586	0.538	**0.221**
FID-TOD w/ API-descriptions (N=1)	**1.653**	0.896	0.658	0.543	0.218
FID-TOD w/ API-descriptions (N=5)	1.655	**0.897**	**0.663**	0.544	0.219
FiD-TOD NoStack	1.683	0.895	0.653	0.543	0.215
FiD-TOD	1.676	**0.897**	0.660	**0.545**	0.217

Table 6: Results on SGD development set. By default, the retrieval module retrieves Top-5 slot information entries from the cache.

by more than 5%. This suggests that incorporating descriptions can enhance the model's ability to generalize to unseen dialog domains.

API-descriptions When evaluating "API-descriptions", where a single API entry in the cache encompasses all intents and slots information for the whole dialogue. We see that the model has high Top-1 accuracy and Top-5 accuracy. This suggests that the model has a high potential to retrieve all the related intents and slots information with a single retrieval attempt.

Orders and Special Tokens We test with different templates, such as switching orders of intents and slots, and find no significant differences. We also find that adding the special tokens "INTENT" and "SLOT" slightly decreases the Top-1 accuracy.

Negative Sampling Experiments with both normal and hard negative pairs, including varying numbers of hard negative pairs, showed no significant impact on retrieval performance. This could be attributed to the fact that, unlike longer passages in question answering, dialog intents, slots, and APIs are generally easier to distinguish when they are referenced in the dialog context.

5.3 Performance of Variants of Cache on End-to-End TOD

As our design involves several templates for the cache, we aim to assess the impact of various cache templates on the performance of the end-to-end TOD system. As shown in Table 5, we observe that "FiD-TOD" using only names already outperforms "MinT (BART-Large)", and adding descriptions further improves the performance. For instance, the cache template"*INTENT: intent name, service description, intent description, SLOT: slot name, slot description*" surpassess both "MinT (BART-Large)" and "*INTENT: intent name, SLOT: slot name*" by 7.4% and 2.7% in terms of Unseen JGA, respectively.

Influence of Irrelevant Information on the End-to-End TOD Given the potential emergence of unseen intents and slots in real-world scenarios, it is challenging to expect a perfect retrieval module.

In this section, we first investigate the ability of the TOD to ignore irrelevant retrieved information. In this section, we first investigate "*the TOD's ability in learning to ignore irrelevant retrieved information*". Table 6 shows the corresponding results. As described in Sec 3.1, "*API-descriptions*" includes all intents and slots for the whole dialogue. The retrieval module exhibits an 84.4% Top-1 accuracy in retrieving all slot information in a single attempt, as shown in Table 4. However, when setting N to 5, the retriever returns similar yet irrelevant information despite a near 100% Top-5 recall accuracy. This results in the inclusion of lots of irrelevant intents and slots into the generative model. Interestingly, "*API-descriptions (N=1)*" performs similar to *API-descriptions (N=5)*, suggesting that the TOD is capable of learning to ignore irrelevant retrieved information.

Second, we investigate "*if the TOD generator*

...	...
SYSTEM:	Do you want to make a reservation for 2 people in the restaurant?
USER:	Yes, thanks. What's their phone number?
RETRIEVAL: (Predicted Top-5)	INTENT: ReserveRestaurant , a popular restaurant search and reservation service , make a table reservation at a restaurant , SLOT: number_of_seats , number of seats to reserve at the restaurant INTENT: ReserveRestaurant , a popular restaurant search and reservation service , make a table reservation at a restaurant , SLOT: time , tentative time of restaurant reservation INTENT: ReserveRestaurant , a popular restaurant search and reservation service , make a table reservation at a restaurant , SLOT: date , tentative date of restaurant reservation INTENT: ReserveRestaurant , a popular restaurant search and reservation service , make a table reservation at a restaurant , SLOT: restaurant_name, name of the restaurant INTENT: ReserveRestaurant, a popular restaurant search and reservation service , make a table reservation at a restaurant , SLOT: location , city where the restaurant is located
APICALL: (Gold)	api_name = ReserveRestaurant ; date = 2019-03-01 ; location = San Jose ; number_of_seats = 2 ; restaurant_name = Sino ; time = 11:30
APICALL: (Predicted)	api_name = ReserveRestaurant ; date = 2019-03-01 ; city = San Jose ; party_size = 2 ; restaurant_name = Sino ; time = 11:30
APIRESP:	city = San Jose ; cuisine = Asian ; has_live_music = False ; phone_number = 408-247-8880 ; price_range = moderate ; restaurant_name: Sino; serves_alcohol = False ; street_address = 377 Santana Row
SYSTEM:	The phone number is 408-247-8880.

Table 7: A predicted example on the development set. Red colors indicate incorrect predictions and light blue colors indicate correct slots.

relies more on the retriever when all retrieved information entries are stacked together". To this end, we compare "FiD-TOD" and "FiD-TOD NoStack", with the difference being whether the retrieved information entries are handled collectively or separately. As shown in Row 3 of Table 6, "FiD-TOD NoStack" performs slightly worse when not stacking all retrieved information directly with a single dialog context. This could be attributed to the design of "FiD-TOD NoStack", which results in repeated dialog context during each retrieval attempt and may hinder the retrieved information.

Error Analysis Despite the retrieval module demonstrating relatively high Top-5 accuracy, there is still room for improvement in the Joint Goal Accuracy (JGA). Therefore, we examine potential reasons for this discrepancy. Table 7 shows one most frequently appeared error type, where the retrieval module successfully retrieve Top-5 information entries from the cache. In terms of APICALL prediction, the TOD accurately generates the intent and associated values. Among the generated slots,"*city*" and "*party_size*" are semantically similar to "*location*" and "*number_of_seats*", respectively. However, the two generated slots are incorrect as they belongs to different services. Upon further inspection, we find these terms are from the training cache. This suggests that the TOD generator does not completely rely on the retriever, and it tends to memorize the training slot information entries from the training cache, pointing towards the need for better generalized abilities. Furthermore, approximately 20% dialogue turns on the developments set shows this issue, suggesting a huge space to improve the performance. We hypothesize that data augmentation, such as entity replacements in dialog history, could be one possible way to mitigate this problem. We leave further exploration of this issue to future work.

6 Conclusion

This paper aims to improve zero-shot performance of end-to-end TOD systems with a simple cache. We first construct a simple cache with intents and slots and fine-tune a retrieval module to retrieve most relevant information entries. We then train the end-to-end TOD model to reference and ground the dialog history and the retrieved information while performing TOD generation. Experimental results on a large-scale Google Schema-Guided Dialogue dataset show that our approach has superior performance over strong baselines.

References

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. Task-optimized adapters for an end-to-end task-oriented dialogue system. *ACL Findings*.

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

Moya Chen, Paul A Crook, and Stephen Roller. 2021. Teaching models new apis: Domain-agnostic simulators for task oriented dialogue. *arXiv preprint arXiv:2110.06905*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*, pages 4171–4186.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*.

Yihao Feng, Shentao Yang, Shujian Zhang, Jianguo Zhang, Caiming Xiong, Mingyuan Zhou, and Huan Wang. 2023. Fantastic rewards and how to tame them: A case study on reward learning for task-oriented dialogue systems. *ICLR*.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. Neural approaches to conversational ai. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 2–7.

Silin Gao, Ryuichi Takanobu, Wei Peng, Qun Liu, and Minlie Huang. 2021. Hyknow: End-to-end task-oriented dialog modeling with hybrid knowledge management. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1591–1602.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using gpt-2. In *ACL*, pages 583–592.

Wanwei He, Yinpei Dai, Min Yang, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022a. Unified dialog model pre-training for task-oriented dialog understanding and generation. In *SIGIR*, pages 187–200.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, et al. 2022b. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. *AAAI*.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *NeurIPS*, 33:20179–20191.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. *EACL*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Mojtaba Komeili, Kurt Shuster, and Jason Weston. 2021. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*.

Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. 2022. Sgd-x: A benchmark for robust generalization in schema-guided dialogue systems. In *AAAI*, volume 36, pages 10938–10946.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. Mintl: Minimalist transfer learning for task-oriented dialogue systems. In *EMNLP*, pages 3391–3405.

Qi Liu, Lei Yu, Laura Rimell, and Phil Blunsom. 2021. Pretraining the noisy channel model for task-oriented dialogue. *Transactions of the Association for Computational Linguistics*, 9:657–674.

Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh, and Jason Weston. 2017. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

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.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Buildingtask bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020a. Schema-guided dialogue state tracking task at dstc8. *arXiv preprint arXiv:2002.01359*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020b. Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset. *AAAI*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020c. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2021. Multi-task pre-training for plug-and-play task-oriented dialogue system. *arXiv preprint arXiv:2109.14739*.

Tian Xie, Xinyi Yang, Angela S Lin, Feihong Wu, Kazuma Hashimoto, Jin Qu, Young Mo Kang, Wenpeng Yin, Huan Wang, Semih Yavuz, et al. 2022. Converse–a tree-based modular task-oriented dialogue system. *arXiv preprint arXiv:2203.12187*.

Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2. *AAAI*.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117.