Towards Building Accurate End-to-End Task-Oriented Dialog Systems with a Simple Cache

Anonymous ACL 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 higher flexibility with a simple cache. The cache provides the flexibility to dynamically update the TOD systems to disable existing or add new and unseen domains, intents, slots, etc., without intensive retraining. Towards this end, we first fine-tune a retrieval module to retrieve Top-N slot information entries from the cache correctly and then train generative end-to-end TOD models with the cache. While performing TOD generation, the models could refer to and ground on both dialog history and the retrieved information. The introduced cache is easy 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 proposed end-to-end framework over baselines, e.g., the Non-Empty JAG is improved by 6.67% when compared with BART-Large.

1 Introduction

Task-oriented dialog (TOD) systems, as assistant tools in various tasks such as restaurants booking, alarm setting, and recommendations, have played an important role in daily life (Gao et al., 2018; Xie et al., 2022). Currently, the TOD systems can be categorized into two groups: pipeline-based dialog systems, and end-to-end dialog systems. The pipeline-based dialog systems consist of four different modules, *i.e.*, a natural language understanding (NLU) module to detect user intents, a dialog state tracking (DST) module to track the belief states of users across dialog turns, a dialog management (DM) module to carry out the system actions to interact with users based on the dialog states, as well as a natural language generation (NLG) module

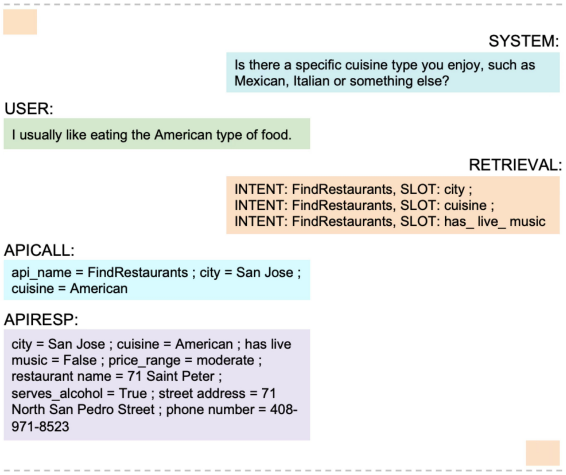

Figure 1: An example of the auto-regressive TOD. We mainly show the APICALL generation process here, and the system response generation process is similar. Here N is set to 3 for the retrieval module.

to generate natural-language responses for users. However, building such pipeline-based systems is label-intensive, error-propagated, and hard to scale (Hosseini-Asl et al., 2020; Peng et al., 2021).

Recently, various approaches have been proposed to generate the dialog states and dialog responses based on seq2seq models, 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., 2022). Compared with the pipeline-based methods on several public datasets, these methods have shown effectiveness, with fewer direct annotations such as user intents and dialog acts. Furthermore, they can easily benefit from the strong capabilities of large-scale pre-trained language models (*e.g.*, GPT-2 (Radford et al., 2019), T5 (Raffel et al., 2019) and BART (Lewis et al., 2020a)) in NLU and NLG. However, they are limited in the flexibility of dynamically handling existing, unseen, or emerging domains, services, slots, etc (Hosseini-Asl et al.,

2020; Peng et al., 2021).

Additionally, another line of work aims to augment the generative models with retrieved information in open-domain question answering and open-domain dialog systems. In particular, these models first retrieve information related to the question or user query from a database of sentences or passages, then incorporate such information into the models to generate answers (Karpukhin et al., 2020; Izacard and Grave, 2021) or open-domain dialog responses (Dinan et al., 2018; Lewis et al., 2020b; Shuster et al., 2021).

In this paper, we resort to combining the both worlds into end-to-end TOD systems. We train the end-to-end TOD models by introducing a simply designed cache, which contains all the accessible intents, slots, corresponding descriptions, or other information from the available dataset. Our approach can reference and ground the retrieved information from the cache while performing TOD. Figure 1 shows a motivating example of our approach, in which the RETRIEVAL module retrieves slot information entries from the cache. APICALL is identical to the dialog states. APIRESP returns information from external API interactions between the system and system databases. The cache could be decoupled from the actual inference, and it has the flexibility to be dynamically updated to add new domains, intents, etc., without intensive retraining. The flexibility would also be helpful to find and disable the wrong predicted slots in the future.

To build an accurate end-to-end TOD system with a simple cache, we first fine-tune a retrieval module to retrieve Top-N slot information entries from the cache correctly. Then we plug it into generative models to perform end-to-end TOD generation. We conduct experiments on the large-scale schema-guided dialog dataset (SGD) (Rastogi et al., 2020a) with around half unseen domains and services to validate the effectiveness of our approach.

The contributions of this paper are as follows: (1) We design a simple yet effective end-to-end TOD framework with a simply designed cache. The backbone models are compatible with existing pre-trained generative models. (2) Our approach provides the flexibility to dynamically update the cache to disable existing or add new or unseen domains, slots, APIs, etc., without intensive retraining. (3) Experimental results demonstrate the superior performance of our approach when compared with strong baselines. (4) We conduct extensive ablation studies and analyses to facilitate further research on building accurate end-to-end TOD systems.

2 Related Work

Dialog Systems There are two types of dialog systems. The first type is the chit-chat system (i.e., chatbot), which engages with users (Roller et al., 2020; Xie et al., 2022). Various chatbots with personalization (Zhang et al., 2018), empathy (Ma et al., 2020) and grounded images (Shuster et al., 2020) are also widely studied. The second type is the task-oriented dialog system (TOD), which assists users in completing specific tasks such as alarms setting and reservations making (Gao et al., 2018; Xie et al., 2022). One typical TOD is pipeline-based, which has individual modules, including NLU (Zhang et al., 2020a; Xu et al., 2021), DST (Zhang et al., 2020b; Qian et al., 2021), DM (Schatzmann et al., 2007; Takanobu et al., 2019), and NLG (Kale and Rastogi, 2020; Su et al., 2020). Although the pipeline-based TOD systems are shown to be effective, they are hindered by the issue of error propagation from one module to the subsequent modules, and the difficulty of deployment at a large scale in practice.

End-to-End TOD Systems Recently, end-to-end TOD models (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., 2022) have shown promising performance on public datasets. From our investigation, these approaches for TOD systems generally follow the following patterns: (1) Rely on powerful pre-trained seq2seq models. (2) Directly use language modeling objective to generate NLU and NLG, or add auxiliary multi-task goals such as DST loss. (3) Directly fine-tune models on the target dataset or conduct pre-training on several TOD dialogue datasets first. (4) Design or adopt data augmentation techniques such as back-translation and entity replacement to augment training dialogues due to the laboriousness to collect many TOD dialog corpora. Specifically, Hosseini-Asl et al. (2020) design a simple language model for TOD based on the DistilGPT2. They generate user belief states and system responses in an auto-regressive way. Peng et al. (2021) build the model based on GPT-2-small, which includes two auxiliary tasks, i.e., the belief state prediction, and the grounded response generation. Yang et al. (2021) incorporate the inter-

mediate information into the context in the dialog session level, and fine-tune the DistilGPT2 with the language modeling objective. Gao et al. (2021) enable the belief state to interact with both structured and unstructured knowledge. Some other works such as Peng et al. (2021); Su et al. (2021); Liu et al. (2021) pre-train language models on several task-oriented dialog datasets to further improve the performance of TOD systems. Hosseini-Asl et al. (2020) show that the interaction with the system database does not help to improve performance. Moreover, these models lack the flexibility to dynamically update to disable existing or add new emerging domains and services.

Retrieval-Augmented Models Our work is also related to the retrieval-augmented models that are widely used in open-domain question answering. Karpukhin et al. (2020) propose a BERT-based (Devlin et al., 2019) dual-encoder framework to retrieve passages from Wikipedia. They are also incorporated into open-domain conversations to reduce hallucination and enrich engagement with users (Shuster et al., 2021; Komeili et al., 2021). The retrieval-augmented models could 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 previous work, our work integrates retrieval models into end-to-end TOD and constructs a simple cache to enable the TOD with more flexibility. The TOD can dynamically adjust the cache of available slots to disable existing or add new domains, intents, etc., without much retraining.

3 TOD Systems with a Simple Cache

Here we present the end-to-end framework with a simple cache, where we build our framework based on BART. In our framework, several generative models such as GPT-2 and T5 could also be directly used as backbone models. Generally, our framework comprises two parts: the first part is a retrieval model to retrieve slot information entries from the cache; the second part is an end-to-end TOD that generates APICALLs and system responses based on the dialog history and the retrieved slot information. Figure 2 demonstrates one simple variant of our framework, *i.e.* "BART+Retriever", where the retrieved slot information entries are stacked together. We also design the other variant (*i.e.*, "FiD-TOD") in Sec. 4.2, where each retrieved slot is concatenated with the dialog history and then all the information are concatenated together and sent to the decoder. Figure 3 shows the "FiD-TOD" model. To better illustrate the framework, we list the simpler "BART+Retriever" in the main figure and this section.

3.1 Construction of Simple Cache

As aforementioned, a cache of slots or other information can benefit models' reference and ground procedure when performing TOD generation. To this end, we first build a simple cache, which includes names and corresponding descriptions of intents and slots from accessible dialogues. Explicitly, it incorporates all the intents and slots information from training dialogues during training process, and it includes corresponding information from the validation/test set in the validation/test phase. Table 1 illustrates several templates for the cache construction. Specifically, we design "latest API-description" as one type of cache. It includes all the intents and relevant slots in a whole dialogue; as such, it consists of redundant information because when the conversation is not finished, a dialogue turn may only involves a few active intents and slots, while the others are not active. Thus it could help us test whether the model could learn to ignore irrelevant information. Except the typical listed templates in Table 1, we also design several other templates with special tokens such as "*[INTENT] intent name [SLOT] slot name*", as well as those with different orders of intents and slots such as "*intent name, intent description, slot name, slot description*" and "*intent name, slot name, intent description, slot description*". We also discuss the effects of different types of cache in the experimental part.

3.2 Retrieval Module

Given a dialog history c, the TOD system will trigger a retrieval module to retrieve Top-N slots $s_1, \ldots, s_N$ from the cache. In detail, based on the user dialog history until the most recent user turn, the system first calls the retrieval module to generate an APICALL which includes relevant active intents, slots and values. After that, the system continues to trigger the retrieval module to generate a system response based on all previous information.

To correctly retrieve information 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 con-

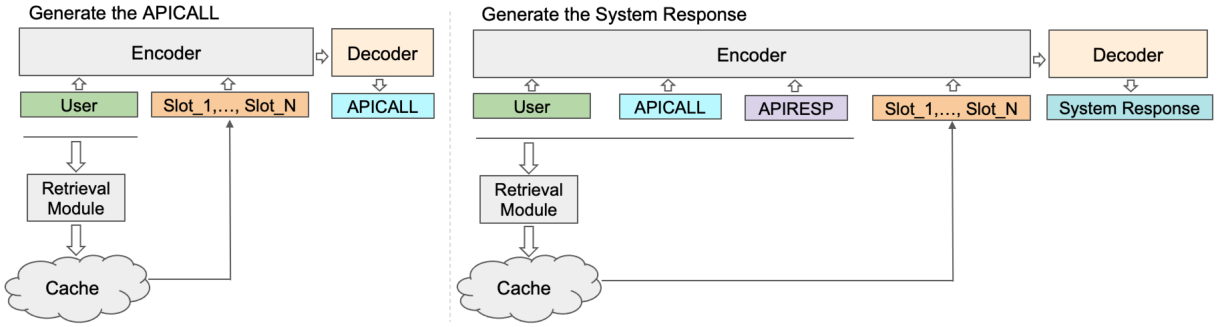

Figure 2: Illustration of the end-to-end framework with a simple cache. The left part is an intermediate step of the right part, indicating the process of generating an APICALL. The right part is the process of producing a system response. In general, based on the dialog history until the user turn, the retrieval module retrieves slots or other information entries from the cache. The dialog history and the retrieved information are sent to the decoder to generate the APICALL. The system continues to retrieve from the cache based on all previous information, and finally, it generates a system response.

Simple 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
latest 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. With regard to the "latest API-descriptions", "api_name" denotes the intents for the whole dialogue, and "optArg" and "reqArg" are all relevant slots in the whole dialogue. Some other templates can be found in Table 4

trastive learning. Specifically, we yield the hidden representation $\mathbf{h}_c$ for the dialog history through an encoder model, *e.g.*, $\mathbf{h}_c = \text{BERT}_c(c)$. We use another BERT encoder to get the feature representation $\mathbf{h}_s$ for a slot information entry of the cache, *i.e.*, $\mathbf{h}_s = \text{BERT}_s(s)$. We measure the similarity between the dialog history and the information entry as: $sim(c, s) = \mathbf{h}_c^T \odot \mathbf{h}_s$. There are n relevant (positive) entries and m irrelevant (negative) entries given one dialog history, where n and m may vary as each dialog history would contain different active slots. We aim to learn a function to minimize the distance between the pairs of relevant dialog history and slot information entries than the irrelevant pairs. Given a specific pair, the corresponding loss function is as follows:

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

Note that we only optimize the retrieval module for generating correct APICALLs. Currently, we do not further fine-tune the retrieval module for generating better system responses, and we leave the optimization to our future work.

Negative Sampling In a training batch, slot information entries irrelevant to the dialog history could naturally be formalized as negative pairs. To improve the robustness and performance of the retrieval module, we design some hard negative pairs. In general, we have two ways: The first way is manually combining intents and slots; for instance, given a positive pair "*active intent, active slots*", a natural negative pair could be "*inactive intent, inactive slots* ". We further construct hard negative pairs such as: "*active intent name, inactive slots from the same active intent*" and "*inactive intent which are semantically similar to the active intents, active slots from the active intents*". The second way is automatically select hard negative pairs during the training process. Specifically, we treat Top-K ranking slot entries from the cache but not relevant to the dialog history as hard negative pairs. Currently, we use the first way to conduct negative sampling and leave the second way to future work.

3.3 End-to-End TOD Systems

Our end-to-end TOD framework generates the APICALL and system response in an auto-regressive way. Figure 1 shows an example. As mentioned be-

fore, the APICALL is similar to the dialogue states, and it contains active intent, slots and corresponding slot values (Chen et al., 2021). Given that a dialogue consists of several turns, the TOD framework triggers the retrieval module twice at a specific dialogue turn. The system first retrieves slot information entries from the simple constructed cache, *i.e.*,

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

Then it generates an APICALL, *i.e.*,

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

After that it will continue retrieving another Top-N slot information entries from the cache, *i.e.*,

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

where the APIRESP is automatically yielded through an API interaction with the original system database when we have an active APICALL and it does not need to be predicted.

Finally it generates a system response:

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

4 Experimental Settings

4.1 Dataset

Most existing 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., 2022) focus on the MultiWOZ datasets (Budzianowski et al., 2018; Zang et al., 2020). However, MultiWOZ only has five valid domains and around 35 useful slots, and all the domains and slots of the test set appear in the training set. Instead, we focus on the Google Schema Guided Dialog (SGD) dataset (Rastogi et al., 2020b) to better test the performance of end-to-end TOD systems. SGD includes over 16k multi-domain conversations across more than 16 domains and 200 slots. Specifically, over half of the services are unseen in the test set. Thus it is a good testbed to test the performance of TOD systems on various domains and zero-shot generalization of new services and skills. Table 2 summarizes the statistics of the adopted dataset.

4.2 Models

Our end-to-end TOD framework's backbone models are compatible with most pre-trained generated models, *e.g.*, GPT-2, T5, and BART. These models have been widely applied to settings of previous state-of-the-art approaches such as (Chen et al., 2020; Lin et al., 2020; Su et al., 2021). We follow them and run primary experiments on SGD, and we find that BART performs best compared with other pre-trained models with similar parameters. We thus use the BART-Large model as a backbone and baseline model. The implementation process of BART-sarge model for TOD is similar to that of (Lin et al., 2020; Chen et al., 2021). Besides, previous work (Peng et al., 2021; Su et al., 2021; He et al., 2022) first pre-train models on several task-oriented dialogue datasets or add multi-task learning. We instead only train the model on the SGD dataset, and our goal is to verify our end-to-end TOD systems with a simple cache and not claim state-of-the-art performance.

We fine-tune the retrieval module up to 3 epochs based on open-sourced DPR[1]. We run our experiments based on the ParLAI platform (Miller et al., 2017), where we mainly rely the BART model and the FiD-RAG model[2]. The FiD model and RAG model originally come from (Lewis et al., 2020b; Izacard and Grave, 2021). By default, we use the RAG-Turn Token setting (Shuster et al., 2021) for the RAG model. We design two variants for end-to-end TOD systems: the first variant, termed FiD-TOD, is based on the FiD-RAG model, as shown in Figure 3. The second variant is "BART+Retriever", as illustrated in the previous Figure 2. FiD-TOD is

	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.

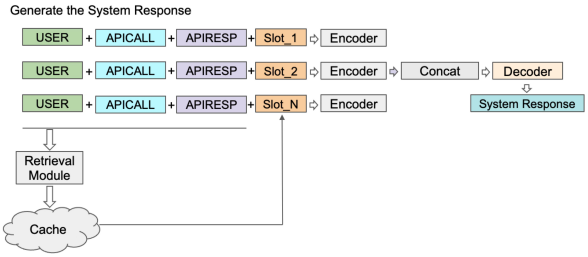

Figure 3: Illustration of the end-to-end FiD-TOD framework with a simple cache. Here we show the system response generation process, and the APICALL generation process is similar to that in Figure 2.

[1] https://github.com/facebookresearch/DPR

[2] https://github.com/facebookresearch/ParlAI/tree/main/parlai/agents/rag

	PPL	JGA	Empty JGA	Non-Empty JGA	Token ACC	Token EM	BLEU-4
BART-Large	2.385	0.8119	**0.9901**	0.3643	0.8365	0.4970	**0.1786**
FiD-TOD	**2.133**	**0.8293**	0.9878	**0.4310**	**0.8393**	**0.5013**	0.1785

Table 3: Testing results on the SGD dataset. The cache of FiD-TOD uses the second template in Table 1.

Database Type	Top-1	Top-2	Top-3	Top-4	Top-5
INTENT: intent name, SLOT: slot name	**0.8326**	**0.8820**	0.9144	0.9447	0.9602
INTENT: intent name, service description, intent description, SLOT: slot name, slot description	**0.8871**	**0.9215**	0.9522	0.9755	0.9803
intent name, slot name, intent description, slot description	**0.8349**	**0.9055**	0.9276	0.9458	0.9548
intent name, slot name, service description, intent description, slot description	**0.9132**	**0.9431**	0.9652	0.9765	0.9806
latest API-descriptions	**0.8438**	**0.9271**	0.9557	0.9623	0.9672

Table 4: Top-5 retrieval accuracy on the test set of SGD. We highlight the Top-1 and Top-2 accuracy.

the same as the "BART+Retriever" model, except that the dialog history first concatenates with each retrieved slot, and then they are sent to the shared encoder. After that, all information are merged and sent to the decoder to generate the APICALL and the system response. The retrieval module and the TOD generator could be trained jointly or separately. In our work, we train them separately as we found it could stabilize the training process and improve performance, when compared with the joint training. Similar findings are also found in open-domain question answering and open-domain conversations (Lewis et al., 2020b; Izacard and Grave, 2021; Shuster et al., 2021).

We retrieve Top-5 slot information entries from the cache and truncate the tokens of dialog history to 256, unless otherwise stated. We align one hard negative pair to each positive pair for the DPR fine-tuning. We fine-tune the end-to-end auto-regressive TOD model up to 4 epochs with an overall batch size of 64 on 8 NVIDIA Tesla V100 GPUs, where each GPU has 16GB memory. The training process takes around 8 hours.

4.3 Evaluation Metrics

We evaluate the end-en-end TOD framework using the following metrics: (1) Top-N accuracy: It 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 (JGA): It evaluates whether the predicted APICALL 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) Empty JGA: As many dialogue turns do not involve an APICALL, *i.e.*, the APICALL is empty, we employ Empty JGA to evaluate whether the model should trigger an API retrieval. (4) Non-Empty JGA: It evaluates whether JGA is correct if the model calls the API. As most dialogue turns have empty APICALL and identifying Empty JGA is relevantly easy, we treat Non-Empty JGA as the most crucial metric for JGA. (5) Token EM: It evaluates the utterance-level token accuracy. Roughly corresponds to perfection under greedy search (generative only). (6) Perplexity (PPL) and Token ACC: Both measure the generative model's ability to predict individual tokens. PPL measures perplexity, and Token ACC measures the per-token accuracy. (7) BLEU-4: It measures the BLEU score (Papineni et al., 2002) between the predicted system response and the reference response. [3]

5 Experimental Results

End-to-End TOD Performance Table 3 shows the overall performance on the test set. FiD-TOD outperforms BART-Large on most metrics. Specifically, it improves the essential NLU metric, *i.e.*, Non-Empty JGA, by 6.67%. The other metrics related to NLG are also slightly improved. We emphasize that due to GPUs resources limits at the later stage of the project in school, we only fine-tune the retrieval module to retrieve correct slot information entries for the APICALL generation. We did not optimize the retrieval module to retrieve slot information entries for the system response generation, *i.e.*, there are noises introduced to the system response generation. We conjecture that we could continue to improve NLG performance with

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

	PPL	JGA	Empty JGA	Non-Empty JGA	Token ACC	Token EM	BLEU-4
BART-Large	1.700	0.8760	0.9918	0.5864	0.8754	0.5379	0.2205
INTENT: intent name, SLOT: slot name	1.688	0.8891	0.9914	0.6329	0.8710	0.5384	0.2121
intent name, slot name, intent description, slot description	1.679	0.8947	0.9881	0.6610	0.8716	0.5409	0.2145
intent name, slot name, service description, intent description, slot description	1.679	0.8939	0.9918	0.6490	0.8713	0.5409	0.2116
INTENT: intent name, service description, intent description, SLOT: slot name, slot description	1.676	0.8968	0.9913	0.6604	0.8726	0.5440	0.2172

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

	PPL	JGA	Empty JGA	Non-Empty JGA	Token ACC	Token EM	BLEU-4
BART-Large	1.700	0.8760	0.9918	0.5864	0.8754	0.5379	0.2205
Latest API-descriptions (N=1)	1.653	0.8963	0.9914	0.6584	0.8735	0.5428	0.2178
Latest API-descriptions (N=5)	1.655	0.8973	0.9908	0.6634	0.8739	0.5450	0.2192
BART + Retriever	1.683	0.8945	0.9909	0.6533	0.8718	0.5425	0.2145
FiD-TOD	1.676	0.8968	0.9913	0.6604	0.8726	0.5440	0.2172

Table 6: Development results on SGD. By default, the retrieval module will retrieve Top-5 slot information entries from the cache.

a further fine-tuned retrieval module for the system response generation.

Retrieval Performance We hope the model could generalize well since there could be lots of unseen services, domains and slots in real scenarios. Table 4 shows the Top-5 retrieval accuracy on the test set. We can see that the model handles well both seen and unseen intents and slots. It has good Top-1 accuracy and above 96% Top-5 accuracy. Compared to only using names, adding related service and intent descriptions improves the Top-1 accuracy by more than 5%, which indicates that adding descriptions could help the model generalize to new unseen domains and services. Regarding the "latest API-descriptions", a single information entry in the cache includes all intents and slots information for a whole dialogue. We see that the model has good Top-1 accuracy and high Top-5 accuracy, *i.e.*, the model has a high potential to retrieve all the related intents and slots information through single-time retrieval. Besides, we test some other templates, such as switching orders of intents and slots, and we find no significant differences. We also find that adding the special tokens "INTENT" and "SLOT" slightly deteriorates the Top-1 accuracy.

We experiment with both normal negative and hard negative pairs, including different numbers of hard negative pairs, and we do not find significant differences in w.r.t retrieval performance. One potential reason is that the number of candidates in the cache is pretty tiny, *i.e.*, the number of queries (dialog history) is much larger than the number of candidates. Consequently, the DPR model is easy overfitting upon one epoch training.

Performance of Variants of Cache on End-to-End TOD We design various templates for the cache, where the information entries in the cache are retrieved by the retrieval module and incorporated into the end-to-end TOD. We thus want to test the influence of various cache templates on the end-to-end TOD. Table 5 shows the development set results. FiD-TOD with only using names is already much better than BART-Large, and adding descriptions could further improve the performance, *e.g.*, "*INTENT: intent name, service description, intent description, SLOT: slot name, slot description*" outperforms BART-Large and "*INTENT: intent name, SLOT: slot name*" by 7.40% and 2.75% on Non-Empty JGA, respectively.

Influence of Irrelevant Information on the End-to-End TOD As many emerging and unseen services, intents, and slots could appear in real scenarios, and it is thus hard to expect a perfect retrieval module. In this section, first, we investigate "*the TOD's ability in learning to ignore irrelevant retrieved information*". Table 6 shows the corresponding results. Note that "*latest API-descriptions*" includes all intents and slots for the whole dialogue, and the retrieval module has 84.38% Top-1 accuracy to all the information in a single time as shown in Table 4. Recall that Top-5 accuracy measures whether the gold values appear on the Top-5 retrieved information entry candidates. If we set N to 5, the retriever will return similar but irrelevant information, even though the Top-5 accuracy is around 100%. Thus it will incorporate several irrelevant intents and slots. However, we find that "*latest API-descriptions (N=1)*" performs similar to *latest API-descriptions (N=5)*, as shown

...	...
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.

in row 2 and row 3 of Table 6. It indicates that the TOD can learn to ignore irrelevant retrieved information.

Second, we investigate "*whether the TOD generator ground more on the retriever if we stack all retrieved information together*". Here we compare "BART+retriever" and FiD-TOD, and their difference is on whether we handle the retrieved information entries together or separately. Row 3 in Table 6 shows that "BART+retriever" performs slightly worse after stacking the retrieved slots together. This finding indicates that the generator may not be able to ground more on the retriever even if we directly put all the retrieved information into the dialog history.

Error Analysis The retriever module has a reasonably well Top-5 accuracy, whereas the Non-Empty JGA is still far from perfect. We are thus curious what are the potential reasons. Table 7 shows one most frequently appeared error type, where the retrieval module could correctly retrieve Top-5 slot information entries from the cache. Regarding the APICALL prediction, the TOD correctly generates the intent and 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 are from different services, and with further check, we find that they are from the training cache. Which suggests that the TOD generator does not fully trust the retriever, and it just memorizes the training slot information entries in the training cache and requires more generalized ability. Moreover, we found there are around 20% dialogue turns with this issue on the development set, indicating a huge space to further improve the performance. We conjecture that data augmentation, such as entity replacements on dialog history, could be one possible way to mitigate this issue. Another possible way is dynamically updating the training cache during the training process. We leave more study as the future work.

Limitation Our approach needs better retrieval modules to handle very limited intents, slots, etc. The retrieval module is easy to overfit in such scenarios. In addition, our approach requires further designs for both the cache and the framework to integrate retrieved information from the large-scale cache, *e.g.*, the internet or large-scale knowledge bases, into the end-to-end TOD systems.

6 Conclusion

This paper proposes to build accurate end-to-end task-oriented dialog systems with a simple cache. We first construct a simple cache with intents, slots, etc., and fine-tune a retrieval module to retrieve Top-N slot information entries. We then train the end-to-end TOD, where the TOD can ground and reference the dialog history and the retrieved information while performing TOD generation. Experimental results on a large-scale Google Schema Guide Dialog dataset show that our end-to-end TOD framework has superior performance and zero-shot generalized ability.

References

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*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607.

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*.

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, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, Luo Si, et al. 2022. 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. In *EACL 2021-16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 874–880. Association for Computational Linguistics.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520.

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*.

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.

Yukun Ma, Khanh Linh Nguyen, Frank Z Xing, and Erik Cambria. 2020. A survey on empathetic dialogue systems. *Information Fusion*, 64:50–70.

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.

Kun Qian, Ahmad Beirami, Satwik Kottur, Shahin Shayandeh, Paul Crook, Alborz Geramifard, Zhou Yu, and Chinnadhurai Sankar. 2021. Database search

results disambiguation for task-oriented dialog systems. *arXiv preprint arXiv:2112.08351*.

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. Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset. *AAAI*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020b. 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.

Stephen Roller, Y-Lan Boureau, Jason Weston, Antoine Bordes, Emily Dinan, Angela Fan, David Gunning, Da Ju, Margaret Li, Spencer Poff, et al. 2020. Open-domain conversational agents: Current progress, open problems, and future directions. *arXiv preprint arXiv:2006.12442*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152.

Kurt Shuster, Samuel Humeau, Antoine Bordes, and Jason Weston. 2020. Image-chat: Engaging grounded conversations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2414–2429.

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.

Shang-Yu Su, Chao-Wei Huang, and Yun-Nung Chen. 2020. Towards unsupervised language understanding and generation by joint dual learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 671–680.

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*.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 100–110.

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*.

Keyang Xu, Tongzheng Ren, Shikun Zhang, Yihao Feng, and Caiming Xiong. 2021. Unsupervised out-of-domain detection via pre-trained transformers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1052–1061.

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.

Jianguo Zhang, Kazuma Hashimoto, Wenhao Liu, Chien-Sheng Wu, Yao Wan, S Yu Philip, Richard Socher, and Caiming Xiong. 2020a. Discriminative nearest neighbor few-shot intent detection by transferring natural language inference. In *EMNLP*, pages 5064–5082.

Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, S Yu Philip, Richard Socher, and Caiming Xiong. 2020b. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 154–167.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*.