Effective and Efficient Conversation Retrieval for Dialogue State Tracking with Implicit Text Summaries

Seanie Lee†∗ Jianpeng Cheng‡ Joris Driesen‡ Alexandru Coca♣∗ Anders Johannsen‡
†KAIST ‡Apple ♣University of Cambridge
lsi-family02@kaist.ac.kr
‡{jianpeng.cheng, joris_driesen, ajohannsen}@apple.com
♣ac2123@cam.ac.uk

Abstract

Few-shot dialogue state tracking (DST) with Large Language Models (LLM) relies on an effective and efficient conversation retriever to find similar in-context examples for prompt learning. Previous works use raw dialogue context as search keys and queries, and a retriever is fine-tuned with annotated dialogues to achieve superior performance. However, the approach is less suited for scaling to new domains or new annotation languages, where fine-tuning data is unavailable. To address this problem, we handle the task of conversation retrieval based on text summaries of the conversations. A LLM-based conversation summarizer is adopted for query and key generation, which enables effective maximum inner product search. To avoid the extra inference cost brought by LLM-based conversation summarization, we further distill a light-weight conversation encoder which produces query embeddings without decoding summaries for test conversations. We validate our retrieval approach on MultiWOZ datasets with GPT-Neo2.7B and LLaMA-7B/30B. The experimental results show a significant improvement over relevant baselines in few-shot DST settings.

1 Introduction

Dialogue state tracking (DST) is one of the most crucial components in task-oriented dialogue systems. The goal of DST is to track users’ intents, slots and values at every turn of a dialogue based on a predefined schema (Budzianowski et al., 2018). The challenge of training a supervised DST model lies in the cost of dialogue state annotations, which is not scalable to new schemas, domains or annotation languages. To address these challenges, recent works (Hu et al., 2022; Chen et al., 2023) adopt in-context learning with pre-trained large language models (LLM) for few-shot DST. In the few-shot setting, similar dialogue exemplars are retrieved based on the test sample and then these exemplars are added to the LLM prompt for target generation. This approach is attractive since no domain-specific fine-tuning is required for the LLM but it can still generalize to unseen domains.

One challenge in few-shot DST is how to retrieve salient conversation exemplars (e.g., in a set of 3 to 5) from the support set, which serves as demonstrations for the LLM. Ideally, a retrieved exemplar should carry both the same dialogue history and state change as the test sample. However, in a practical few-shot setting (e.g., with at most 100 annotated support examples), it is likely that no exemplar in the support set satisfies the above requirement. Consider a test example with two user turns:

user: book a flight to London Heathrow
system: where are you departing from
user: Amsterdam

It is possible that the closest exemplar we can get from the support set is:

user: I’m leaving Manchester by air
system: where are you flying to
user: To Paris

which neither matches the test dialogue state nor the state change. Nevertheless, we hope that LLM can generalize by learning from such exemplars with an identical user intent. The retrieval task gets harder when the conversation becomes lengthy with only partial history related to the current user input. For example, in another test dialog:

user: what’s the weather in London
system: sunny
user: book a flight to London Heathrow
system: where are you departing from
user: Amsterdam

the user’s current intent is identical to the earlier test sample, but it involves unrelated history. Still we want to match the test sample to a similar exemplar, which reflects the user’s intent up to the current point of conversation. This retrieval cannot be easily accomplished with pre-trained dense
retrieval models based on word or sentence similarity. To optimize retrieval performance, previous works (Hu et al., 2022; Chen et al., 2023) fine-tune a dense retriever with “structurally similar” dialogue examples identified from dialogue state annotations with heuristics. Hu et al. (2022) additionally report that including dialogue state information in the retrieval key is helpful. However, the approach is not scalable to a practical few-shot setting (with fewer than 100 annotated support examples), as fine-tuning easily leads to overfitting and catastrophic forgetting (McCloskey and Cohen, 1989; Lee et al., 2022). It is also impractical to expect every domain owner to create their own fine-tuning data with well-engineered rules.

In this work, we propose a new solution for conversation retrieval starting with the introduction of a LLM-based conversation summarizer. For each exemplar to be indexed and also each test dialog, the summarizer produces a text summarizing what the user wants at this point of the conversation. In Section 2, we provide a discussion of this specific summarization choice and how it compares to dialogue state. The summaries are then used as condensed search keys and queries applicable to pre-trained dense retrievers with standard nearest neighbor search. We empirically show that in the few-shot setting, using summaries as retrieval keys and queries is more effective than using raw dialogues.

Notably the conversation summarization task described above can be easily handled by state-of-the-art LLMs via prompt learning, as we will show in an ablation study. However, the deployment of such a retrieval system also introduces extra model parameters and inference cost. Unlike search keys, which can be pre-built offline, a search query needs to be auto-regressively decoded for each test dialogue right during inference. To improve the efficiency of this conversation retriever, our second contribution in this work focuses on distilling a light-weight conversation encoder which embeds a raw dialogue directly into a vector space similar to the embedding of its summary. The light-weight conversation encoder enables efficient conversation search over a vector database without explicit query generation. When evaluated on the MultiWOZ dataset with GPT-Neo-2.7B (Black et al., 2022), LLaMA-7B, and LLaMA-30B (Touvron et al., 2023) for few-shot DST, we find that the distilled conversation encoder is not only more efficient, but also more effective than a cascaded conversation retriever with explicit query generation. Our approach also significantly outperforms relevant baselines, which use annotated dialogues for retriever fine-tuning.

2 Conversation Retrieval with Summaries for LLM-based DST

In the context of task-oriented dialogues with multiple turns of interactions between a user and a system, the objective of DST is to predict the accumulated intents, slots and values at each user turn. In a LLM-based approach, the generation of a dialogue state is conditioned on a task-specific prompt. The prompt includes at least the test conversation and a set of \( k \) demonstration examples, from which we expect the LLM to learn to generalize. Considering the size limit of the prompt, \( k \) is expected to be small (3-5 examples). Each of the retrieved examples is an annotated conversation sharing similar features as the test conversation. We expect to retrieve these exemplars with a dense retriever from a “support set” (e.g., 100 annotations) that can be constructed with minimal effort for domain scaling.

There are two major challenges of conversation retrieval for LLM-based DST described above. First, a good representation of search keys and queries need to be found. As we analyze in Section 1, the similarity of two dialogues is not directly quantifiable by semantic distance, but rather requires more sophisticated structural matching mechanism or a higher-order similarity function. This requirement leads to the second challenge as to how to train an effective conversation retriever that can scale across domains. Previous works (Hu et al., 2022; Chen et al., 2023) mainly fine-tune pre-trained dense retrievers with annotated dialogues obtained from the support set. However, fine-tuning is not realistic in a few-shot setting and for every domain.

As shown in Figure 1-(a), our work introduces a query/key generation step in the LLM-based DST. The generation is performed with another LLM which transforms the raw dialogue context into a text summary whose similarity can be evaluated more easily with pretrained retrievers. Specifically, the text summary represents the user’s intent up to the current point of the conversation. It grounds the latest user input onto the dialogue history, keeping only information related to the
current user intent. Note that the summary is a contextual rewriting of the current user intent that is possibly expressed in multiple turns, with applied ellipsis recovery (Hardt, 1997) and co-reference resolution (Pradhan et al., 2012). Examples of the conversation summary are in Table 6. The summary can also be viewed as a text description of an updated dialogue state which is to be predicted by the LLM. Unlike the dialogue state, the summary does not maintain all conversation history but only includes information relevant to the current user input.

With the introduction of an explicit query/key generation step, we expect that the conversation retrieval becomes easier and the search index can be built more efficiently. To construct the search index, an offline process can be triggered to generate text summaries for every example in the support set. Note that search key generation does not add any inference cost. However, the query generation step comes at an extra cost since the generation needs to happen in an online process. In the next section, we describe how to make the conversation retrieval more efficient by stepping away from explicit query generation.

3 Conversation Encoder Distillation

Note that in the proposed conversation retriever, the LLM-based conversation summarizer needs to be invoked for every test sample to generate the search query as shown in Figure 1-(a). To eliminate the extra inference cost, we propose to distill a light-weight conversation encoder which directly embeds a dialogue into a vector space similar to its summary, by maximizing their embedding similarity. The encoder is trained with large-scale dialogue-summary pairs generated by the conversation summarizer in an offline process. After training the model, as shown in Figure 1-(b), we can directly encode each dialogue into a query embedding for maximum inner product search. We call our conversation encoder CONVERSE, stand-

ing for CONversation embeddings for VErsatile Retrieval with implicit SummariEs. Next we explain the structure and training objective of CONVERSE.

3.1 Model

Preliminaries In our problem setup, we are given a set of unlabeled conversations between a user and system, denoted as \( D^n = \{x_i\}_{i=1}^n \) where each conversation \( x_i \) consists of \( l_i \) utterances \( u_{i,j}, \ldots, u_{i,l_i} \) and each utterance \( u_{i,j} \) is a sequence of \( T_{i,j} \) tokens \( (x_{i,j,1}, \ldots, x_{i,j,T_{i,j}}) \). As shown in Figure 2-(b), the training data of CONVERSE is prepared by invoking the conversation summarizer to generate a summary for each conversation \( x_i \), denoted as \( z_i \), which consists of \( T_i' \) tokens with \( T_i' \ll \sum_{j=1}^{T_i} T_{i,j} \). We denote the dataset augmented with summaries as \( D^n = \{(x_i, z_i)\}_{i=1}^n \). For brevity, we omit the first subscript \( i \) if there is no ambiguity. Given the set of conversation-summary pairs, the goal is to train an encoder \( f_0 : V^T \rightarrow \mathbb{R}^{T \times d} \) such that the similarity between a conversation and its summary is maximized, where \( V \) denotes a set of predefined tokens.

Conversation and Summary Embedding To match a conversation against a summary, we leverage the commonly used architecture in dense retrieval known as the dual encoder (Yih et al., 2011; Lee et al., 2019; Karpukhin et al., 2020a), where a conversation and a summary are encoded jointly for similarity comparison. State-of-the-art dual encoders (Khattab and Zaharia, 2020) represent each encoding as multiple vectors, typically the contextualized token vectors, to represent the text. These models largely improve the model expressiveness, and exhibit much stronger performance and robustness compared to their single-vector counterparts (Thakur et al., 2021). Based on it, we represent both the conversation embedding \( f_0(x) \) and the summary embedding \( f_0(z) \) as a matrix. While the summary encoder in the dual architecture can be directly integrated into off-the-shelf sentence encoders, our conversation encoder (CONVERSE) is designed to reflect the inductive bias of the summarization task.

CONVERSE Remember the task of the conversation summarizer is to summarize the current user intent by grounding it to the conversation history. Hence the latest user input (the state delta) is most important and any past utterances irrelevant to the latest input should be dropped out.
To reflect the nature of the summarization task, we explicitly model the grounding step between the latest user input \( u_t \) and past utterances \( u_1, \ldots, u_{t-1} \) as a structural bias in CONVERSE. This is achieved with the introduction of a soft retrieval structure that softly retrieves past utterances or tokens which are relevant to the latest user input. Specifically, the soft retrieval is simulated with another neural network \( g_\theta : \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1] \), which outputs the relevance score of each token in the utterances \( u_1, \ldots, u_{t-1} \) conditioned on the latest user utterance \( u_t \). Then, the relevance scores are used to downweight irrelevant token representations of the conversation \( x \):

\[
\hat{f}_{\theta, \phi}(x) = \left[ \begin{array}{c} w_{1,1} f_\theta(x)_{1,1}^\top \\
\vdots \\
 w_{l-1,T_{l-1}} f_\theta(x)_{l-1,T_{l-1}}^\top \\
 f_\theta(x)_{l,1}^\top \\
\vdots \\
 w_{l,T_l} f_\theta(x)_{l,T_l}^\top \\
\end{array} \right] \in \mathbb{R}^{T \times d}
\]

\[
w_{j,t} = g_\phi(f_\theta(x)_{j,t}, s_t(x)) \in [0, 1] \quad (1)
\]

\[
s_t(x) = \frac{1}{T_t} \sum_{l=1}^{T_t} f_\theta(x)_{l,t} \in \mathbb{R}^d,
\]

where \( t \in \{1, \ldots, T_t\} \) for each \( j \in \{1, \ldots, l-1\} \) and \( f_\theta(x)_{j,t} \) is a contextual representation of \( t \)-th token in \( u_j \). Intuitively, an irrelevant token in the conversation history receives a small weight, reducing its contribution to the final similarity scoring against the summary. Conversely, a token in the latest user input always carries the highest weight 1 and contributes more to the similarity computation.

### 3.2 Training Objective

Given the conversation encoder \( \hat{f}_{\theta, \phi} \) and summary encoder \( f_\theta \) with a set of conversation and summary pairs \( \mathcal{D}^a = \{(x_i, z_i)\}_{i=1}^m \), as illustrated in Figure 2-(b), we train the dual encoder to maximize the similarity between a dialogue and its summary with the contrastive loss (Henderson et al., 2017):

\[
L(\theta, \phi; \mathcal{D}^a) = -\frac{1}{n} \sum_{(x, z) \in \mathcal{D}^a} \log p_{\theta, \phi}(z|x) \quad (2)
\]

\[
p_{\theta, \phi}(z|x) = \frac{\exp(\text{sim}(\hat{f}_{\theta, \phi}(x), f_\theta(z)))}{\exp(\sum_{(x', z') \in \mathcal{D}^a} \text{sim}(\hat{f}_{\theta, \phi}(x'), f_\theta(z')))}
\]

where \( \text{sim} \) is the multi-vector similarity function (Khattab and Zaharia, 2020), which computes the similarity between the conversation and its summary, denoted as \( \text{sim}(\hat{f}_{\theta, \phi}(x), f_\theta(z)) \), by averaging maximum dot product between summary tokens and each conversation token as:

\[
\frac{1}{T} \sum_{j=1}^{T} \max_{l=1, \ldots, l'} \frac{\hat{f}_{\theta, \phi}(x)_{j,l}^\top f_\theta(z)_{l'}}{l'}
\]

In practice, due to computational costs, we sample a mini-batch \( B \subset \mathcal{D}^a \) for computing the denominator of the contrastive loss in equation 2.

### 3.3 Inference

In LLM-based DST, we are given a small support set of labeled dialogues \( \mathcal{D}_1^s = \{(x_i^s, y_i^s)\}_{i=1}^m \). The search keys can be pre-built offline by calling the conversation summarizer to generate a summary for each dialogue \( x_i^s \) from the support set \( \mathcal{D}_1^s \), resulting in a set of (conversation, label, and summary) triplets denoted as \( \mathcal{D}_2^s = \{(x_i^s, y_i^s, z_i^s)\}_{i=1}^m \). The search index is then built with the summary as the
key, and a labeled conversation as the value. The summaries are encoded with the fine-tuned summary encoder described in Section 3.2.

During inference, for each conversation \(x^j\) from the test set \(D^q = \{x_i^q\}_{i=1}^Q\), we embed the conversation with the CONVERSE encoder and compute its similarity with every search key using the similarity function in equation 3, i.e., \(\text{sim}(f_{\theta,\phi}(x_i^q), f_{\theta}(z_j^s))\) for \(j = 1, \ldots, m\). As shown in Figure 2-(c), the retrieval ranks examples \((x^j, y^j)\) based on the similarity score and chooses the top-\(k\) exemplars. Finally, the retrieved exemplars are added to the prompt of the downstream LLM for dialog state generation.

4 Experiments

4.1 Experimental Setup

Common We evaluate LLM-based DST with the proposed conversation retriever on MultiWOZ 2.1 (Eric et al., 2020) and 2.4 (Ye et al., 2022).

To simulate few-shot scenario, we consider a support set of 100 labeled conversations as the default setting in our comparison. For each experimental run, we randomly sample 100 labeled conversations from the training data of MultiWOZ 2.1/2.4. The analysis of other support set sizes is deferred to an ablation study. During inference, we retrieve the top 5 examples from the support set. The examples along with a test conversation are inserted into the prompt, following Hu et al. (2022). This setting is applied to all comparisons.

We use both GPT-Neo (Black et al., 2022) and LLaMA-7B/30B (Touvron et al., 2023) as the LLM for DST generation. For evaluation, we report average and standard deviation of Joint Goal Accuracy (JGA) and F1 score (Henderson et al., 2014) on all 7,368 test dialogues from MultiWOZ with three runs.

Baselines We compare the proposed conversation retriever with the following baselines.

1. IC-DST (Hu et al., 2022): It utilizes dialogue labels to construct positive and negative pairs for fine-tuning a pretrained SBERT (Reimers and Gurevych, 2019) or LinkBERT (Yasunaga et al., 2022) as a retriever. The retrieval key is a dialogue context, and the best dialogue context is reported to be previous dialogue state + current user input (which is better than a full dialogue).

2. SM2 (Chen et al., 2023): Similar to IC-DST, it fine-tunes SBERT on labeled dialogue data with contrastive loss, where conversations with partial matching slots or values are considered as positive samples. The retrieval key is a dialogue context similar to IC-DST.

3. GTR-T5-LARGE (Ni et al., 2022): It uses a T5 encoder, which is pretrained on large scale corpora for sentence representation, to compute the similarity between conversations for retrieving examples. The retrieval key is the full dialogue.

4. JINA-LARGE (Günther et al., 2023): Similar to GTR-T5, the pretrained sentence encoder Jina is used to compute similarity between conversations. The retrieval key is also the full dialogue.

Ours We use gpt-3.5-turbo (OpenAI, 2022) as the conversation summarizer, since it provides reliable summaries that satisfy the task requirement in the prompt (see human evaluation in 4.4 and the prompt specified in Appendix A). First, we evaluate the effectiveness of summary-based search key and query generation, using off-the-shelf retrievers GTR-T5-Large and Jina-Large, which are directly comparable with the baseline.

Second, we evaluate the distilled conversation encoder (CONVERSE). To train CONVERSE, we use the same conversation summarizer to generate a summary for every turn of every conversation from the full MultiWOZ training set, resulting in a total of 56,776 conversation-summary pairs. The parameters \(\theta\) of the dual encoder \(f_{\theta}\) and \(f_{\theta,\phi}\) are shared and initialized with LinkBERT (Yasunaga et al., 2022), and trained on the conversation-summary pairs for 20 epochs with the objective in equation 2. LinkBERT (Yasunaga et al., 2022) is chosen since we empirically find that it offers the best general-purpose weight of initialization. We use the AdamW optimizer (Loshchilov and Hutter, 2018) with learning rate \(5\cdot10^{-5}\) and batch size 200. We use eight A100 GPUs for training the model.

4.2 Quantitative Results

Main Results The DST results are shown in Table 1 and Table 2. The first set of comparisons is between conversation retrieval with and without explicit query/key generation. We observe that using the summary as search keys/queries significantly improves the end-to-end (E2E) results, when evaluated with the same off-the-shelf retriever (GTR-T5 or Jina). The result is slightly behind IC-DST which fine-tunes the retriever with dialogue state information in the key. However, after introducing the distilled CONVERSE model, we achieve much
Comparison Against Few-Shot Finetuning

Recently, Mosbach et al. (2023) have shown that few-shot fine-tuning outperforms in-context learning in some settings, which makes people wonder how few-shot fine-tuning behaves with 100 labeled dialogues for dialogue state tracking tasks. To answer this question, we compare our method, CONVERSE, against one of the strongest few-shot fine-tuning methods, DS2 (Shin et al., 2022), using BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) language models. As shown in Table 3, our method, CONVERSE, outperforms the few-shot fine-tuning method, DST. Note that the BART-based DS2 severely overfits to the small labeled dataset and the T5-based model performs worse than the in-context learning method, even though T5 model is pretrained on an additional large-scale labeled dialogue summarization dataset, SAMSum (Gliwa et al., 2019).

Out-of Domain Generalization

To verify our hypothesis that our unsupervised retriever CONVERSE generalizes better to unseen domain than supervised methods, we hold out the hotel domain from the MultiWOZ dataset and train the retrievers, IC-DST and CONVERSE on the remaining four domains: train, restaurant, taxi, and attraction. Then we evaluate the performance of the few-shot in-context learning with the retrievers on test examples from the unseen domain, hotel. As shown in Table 4, our model CONVERSE outperforms IC-DST by a large margin, which empirically validates that our unsupervised retriever generalizes better to unseen domain than the supervised one.

4.3 Ablation Study

Size of Support Set

We empirically study the size of the support set (labeled dialogues) in the conversation retrieval task. Notably, a smaller support set requires less annotation effort from the domain owner, placing more emphasis on general-
Joint Goal Accuracy

Figure 3: JGA of LLaMA-7B with CONVERSE as a function of the number of labeled data.

Table 5: Ablations of different ranking with LLaMA-7B.

<table>
<thead>
<tr>
<th>Model</th>
<th>MultiWOZ 2.1</th>
<th>MultiWOZ 2.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONVERSE + Rerank</td>
<td>19.86 ± 1.22</td>
<td>20.65 ± 1.28</td>
</tr>
<tr>
<td>CONVERSE</td>
<td>19.33 ± 0.91</td>
<td>20.35 ± 1.03</td>
</tr>
</tbody>
</table>

Table 6: The LLM successfully summarize the conversation based on the latest user utterance.

**Summary:** The user wants to book a taxi to be picked up at a specific location and dropped off at another.

**Conversation**

**USER:** I need some tourist information please. I need to know about a hotel called the Arbury lodge guest house.
**SYSTEM:** The Arbury lodge guest house is in the north area and has a moderate price range. · · ·
**USER:** I would like to book a stay for 3 people for 2 nights starting from Tuesday.
**USER:** I am also looking to eat somewhere expensive, in the south area of town.

**USER:** I will also need a taxi, please.
**SYSTEM:** Where would you like your taxi to pick you up and drop you off?
**USER:** I want to be picked up at the hotel and dropped off at the restaurant.

**Summary:** The user wants to book a taxi to be picked up at a specific location and dropped off at another.

**Conversation**

**USER:** I need some tourist information please. I need to know about a hotel called the Arbury lodge guest house.
**SYSTEM:** The Arbury lodge guest house is in the north area and has a moderate price range. · · ·
**USER:** I would like to book a stay for 3 people for 2 nights starting from Tuesday.
**USER:** I am also looking to eat somewhere expensive, in the south area of town.

**USER:** I will also need a taxi, please.
**SYSTEM:** Where would you like your taxi to pick you up and drop you off?
**USER:** I want to be picked up at the hotel and dropped off at the restaurant.

**Summary:** The user wants to book a taxi to be picked up at a specific location and dropped off at another.

**Conversation**

**USER:** I need some tourist information please. I need to know about a hotel called the Arbury lodge guest house.
**SYSTEM:** The Arbury lodge guest house is in the north area and has a moderate price range. · · ·
**USER:** I would like to book a stay for 3 people for 2 nights starting from Tuesday.
**USER:** I am also looking to eat somewhere expensive, in the south area of town.

**USER:** I will also need a taxi, please.
**SYSTEM:** Where would you like your taxi to pick you up and drop you off?
**USER:** I want to be picked up at the hotel and dropped off at the restaurant.

**Summary:** The user wants to book a taxi to be picked up at a specific location and dropped off at another.
null
| Target Conversation           | SYSTEM: There are 9 Indian restaurants in centre what price range do you want?  
USER: I am looking for expensive Indian food.  
| Gold Label | restaurant-food: indian, restaurant-pricerange: expensive  
| Prediction | restaurant-food: indian, restaurant-pricerange: expensive  
| Exemplar #1 | USER: I am also looking to eat somewhere expensive, in the south area of town.  
SYSTEM: There are 2 Chinese, 1 Indian, 1 Italian, and 1 Mexican restaurants. Which of those would you like?  
USER: I would like the Italian place please.  
Summary: The user wants to find a restaurant in a specific area with a certain price range and cuisine.  
Label: restaurant-food: italian.  
| Exemplar #2 | USER: Actually, I also need a moderate priced restaurant in the same area.  
SYSTEM: I can find a few that meet that criteria, would you like Indian, or Italian food?  
USER: Well, everyone said it’s my choice, so I think I would like Italian.  
Summary: The user wants to find a restaurant in a specific area with a specific cuisine.  
Label: restaurant-food: italian.  
| Exemplar #3 | SYSTEM: Good news I was able to get this for you. Reference i4dxhdj1. Can I help you find other things to do in the area as well ?  
USER: I am also looking to eat somewhere expensive, in the south area of town.  
Summary: The user wants to find a restaurant with an expensive price range in a specific area of town.  

Table 8: Given the target conversation, we show the top 3 most similar examples retrieved by our model CONVERSE.

haria, 2020; Izacard et al., 2022; Santhanam et al., 2022) proposing neural network based retrievers which encode queries and keys into low dimensional vectors and compute similarities between them. Hu et al. (2022); Chen et al. (2023) propose to utilize slots and values to represent a long history of conversation for retrieval. However, in order to train the retriever, their approaches require labeled dialogue data to construct positive and negative conversations for each query conversation. Recently, Ravfogel et al. (2023) retrieve texts based on abstract descriptions generated by a LLM.

6 Conclusion

The contribution of this work is twofold. First, we proposed an effective way of retrieving conversations in LLM-based DST with conversation summaries as search keys and queries. We then improved the efficiency of the retrieval system by distilling a conversation encoder capable of embedding a conversation into a vector space similar to its summary. This eliminates the cost of decoding an actual summary for each test sample during inference. We validated our CONVERSE encoder for LLM-based DST in a real few-shot setting with 100 conversations in the support set. Results showed that CONVERSE consistently improved both the efficiency and the performance of few-shot DST when using different LLMs, outperforming previous LLM-based DST baselines that rely on annotated dialogues for retriever fine-tuning.

Acknowledgement We would like to express our sincere gratitude to Dominik Wagner for the numerous helpful pieces of feedbacks.

References


Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li,


Appendix

A Prompt for Summarization

In Table 9, we provide an instruction and two examples that specify how to summarize a conversation based on the latest user utterance. The first example demonstrate how to ignore irrelevant history — request for setting an alarm and focus on sending a message. The second example is to generate “The user wants nothing more” instead of summarizing the conversation.

B Instruction for Human Evaluation

We conduct human evaluation of 135 summaries generated by the conversation summarizer, gpt-3.5-turbo. As shown in Figure 5, three human judges are given an instruction for the evaluation and they are asked to assess whether the generated summaries are consistent with the prompt specified in Table 9.

C Example of Summaries

In Tables 10 to 14, we provide more summaries generated by the LLM, gpt-3.5-turbo.
Instruction

Following is the prompt for summarizing a conversation. Choose yes or no if the generated summaries are consistent with the prompt.

You are shown a conversation between a virtual assistant on a phone and a user. You have to summarise what the user wants at this particular point of the conversation. Your summary should contain the user intent and the slots he mentioned. However, the summary should be a delexicalized abstract sentence, which means it should not contain actual slot values. Note that it is possible that not all conversation history is relevant and you need to summarise based on what is relevant to the most recent user turn.

If the user does not have a goal at this point or his goal gets completed by the system, just summarize that "The user wants nothing more".

---

<fictional_example>

USER: make an alarm for 6
SYSTEM: I have created an alarm at 6
USER: Also, send a message to my wife
SYSTEM: What would you like the message to say?
USER: ehm... happy birthday
SYSTEM: I can do that. What message service do you want to use
USER: whatsapp

What does the user want at this point in the conversation?
The user wants to send a message to a recipient with a given text using a specified app

</fictional_example>

<fictional_example>

USER: make an alarm for 6
SYSTEM: I have created an alarm at 6
USER: thanks you and goodbye

What does the user want at this point in the conversation?
The user wants nothing more

</fictional_example>

Now it's your turn.

(test_example)

What does the user want at this point in the conversation?
You are shown a conversation between a virtual assistant on a phone and a user. You have to summarise what the user wants at this particular point of the conversation. You summary should contain the user intent and the slots he mentioned. However, the summary should be a delexicalized abstract sentence, which means it should not contain actual slot values. Note that it is possible that not all conversation history is relevant and you need to summarise based on what is relevant to the most recent user turn. If the user does not have a goal at this point or his goal gets completed by the system, just summarize that “The user wants nothing more”.

<fictional_example>
USER: make an alarm for 6
SYSTEM: I have created an alarm at 6
USER: Also, send a message to my wife
SYSTEM: What would you like the message to say?
USER: ehm... happy birthday
SYSTEM: I can do that. What message service do you want to use
USER: whatsapp

What does the user want at this point in the conversation?
The user wants to send a message to a recipient with a given text using a specified app
</fictional_example>

<fictional_example>
USER: make an alarm for 6
SYSTEM: I have created an alarm at 6
USER: thanks you and goodbye

What does the user want at this point in the conversation?
The user wants nothing more
</fictional_example>

Now it’s your turn.
{test_example}

What does the user want at this point in the conversation?

Table 9: Prompt with two exemplars for summarizing a conversation.

**Conversation**

**USER:** I am looking for a hotel in the east, and i would like free wifi please.
**SYSTEM:** I have found express by holiday inn Cambridge. 1517 Norman way, Coldhams business park
**USER:** Does that hotel include free parking and have 2 stars?
**SYSTEM:** No, thank you, that s all I need.
**USER:** Sure thing what’s the area and or name?
**SYSTEM:** Thank you for contacting Cambridge Towninfo centre. Have a nice trip! Good Bye!
**USER:** Actually, I also need a moderate priced restaurant in the same area

**Summary:** The user wants to find a moderate priced restaurant in a specific area.

Table 10: Example of conversation summary.
**Conversation**

**USER:** I need some tourist information please. I need to know about a hotel called the Arbury lodge guest house.

**SYSTEM:** The Arbury lodge guest house is in the north area and has a moderate price range. It is 4 stars and has free parking and internet.

**USER:** I would like to book a stay for 3 people for 2 nights starting from Tuesday.

**SYSTEM:** Good news I was able to get this for you. Reference i4dxhdjl. Can I help you find other things to do in the area as well?

**USER:** I am also looking to eat somewhere expensive, in the south area of town.

**SYSTEM:** There are 2 Chinese, 1 Indian, 1 Italian, and 1 Mexican restaurants. Which of those would you like?

**USER:** I would like the Italian place please.

**SYSTEM:** Frankie and Benny's is an expensive restaurant which is expensive. Would you like me to book it?

**USER:** Yeah, book for 3 people on Tuesday at 18:30 and give me the reference number.

**Summary:** The user wants to book a table at an expensive Italian restaurant for 3 people at a specific date and time and receive a reference number.

Table 11: Example of conversation summary.

**Conversation**

**USER:** I would like to get to Cambridge by train, but I am afraid I need to travel on Sunday. Will that be possible?

**SYSTEM:** From where are you departing?

**USER:** I am leaving from Broxbourne.

**SYSTEM:** There are 19 trains running from Broxbourne to Cambridge on Sunday.

**Summary:** The user wants to find a train from a specific location to a specific destination arriving before a certain time.

Table 12: Example of conversation summary.

**Conversation**

**USER:** Please help me reserve a taxi from Saint Catherine's College.

**SYSTEM:** Sure! When would you like to arrive?

**USER:** I would like to arrive by 01:15.

**SYSTEM:** Great! And what is your destination?

**USER:** I am going to Loch Fyne.

**Summary:** The user wants to reserve a taxi from a specific location to a specific destination at a specific time.

Table 13: Example of conversation summary.

**Conversation**

**USER:** Are there any concert halls in the west?

**SYSTEM:** I am sorry there are no listings for concert halls in the west. Is there something else I can assist you with today?

**USER:** What about colleges in the west?

**SYSTEM:** There are 5 colleges in that area. 3 have no entrance fees. Would you like additional information on any of these?

**USER:** No thank you. I am looking for information El Shaddai hotel.

**SYSTEM:** El Shaddai is a guest house in the centre part of town. It has free internet, parking and has 0 stars. Would you like me to book it for you?

**USER:** Could you? That would be great. There are 5 of us and we plan to arrive on Thursday. We’d like to stay for 5 nights.

**Summary:** The user wants to book a hotel for a group of 5 people for a specific duration of time.

Table 14: Example of conversation summary.