A Comparative Study on Schema-Guided Dialogue State Tracking

Jie Cao [*]
School of Computing, University of Utah
jcao@cs.utah.edu

Yi Zhang
AWS AI, Amazon
yizhngn@amazon.com

Abstract

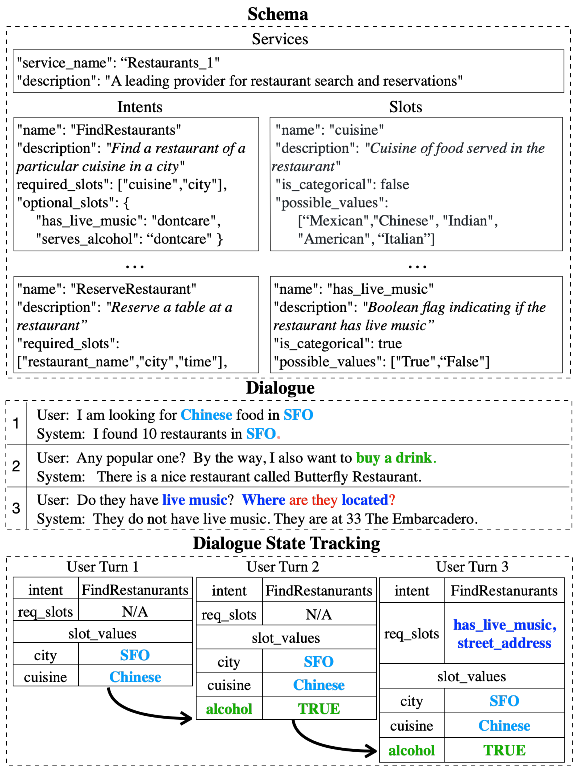
Frame-based state representation is widely used in modern task-oriented dialog systems to model user intentions and slot values. However, a fixed design of domain ontology makes it difficult to extend to new services and APIs. Recent work proposed to use natural language descriptions to define the domain ontology instead of tag names for each intent or slot, thus offering a dynamic set of schema. In this paper, we conduct in-depth comparative studies to understand the use of natural language description for schema in dialog state tracking. Our discussion mainly covers three aspects: encoder architectures, impact of supplementary training, and effective schema description styles. We introduce a set of newly designed bench-marking descriptions and reveal the model robustness on both homogeneous and heterogeneous description styles in training and evaluation.

1 Introduction

From early frame-driven dialog system GUS (Bobrow et al., 1977) to virtual assistants (Alexa, Siri, and Google Assistant *et al.*), frame-based dialog state tracking has long been studied to meet various challenges. In particular, how to support an ever-increasing number of services and APIs spanning multiple domains has been a focal point in recent years, evidenced by multi-domain dialog modeling (Budzianowski et al., 2018; Byrne et al., 2019; Shah et al., 2018a) and transferable dialog state tracking to unseen intent/slots (Mrkšić et al., 2017; Wu et al., 2019; Hosseini-Asl et al., 2020).

Recently, Rastogi et al. (2019) proposed a new paradigm called schema-guided dialog for transferable dialog state tracking by using natural language description to define a dynamic set of service schemata. As shown in Figure 1, the primary motivation is that these descriptions can offer effective knowledge sharing across different services, e.g., connecting semantically similar concepts across heterogeneous APIs, thus allowing a unified model to handle unseen services and APIs. With the publicly available schema-guided dialog dataset (SG-DST henceforward) as a testbed, they organized a state tracking shared task composed of four subtasks: intent classfication (*Intent*), requested slot identification (*Req*), categorical slot labeling (*Cat*), and noncategorical slot labeling (*NonCat*) (Rastogi et al., 2020). Many participants achieved promising performance by exploiting the schema description for dialog modeling, especially on unseen services.

Despite the novel approach and promising results, current schema-guided dialog state tracking



Schema

Services

```
"service_name": "Restaurants_1"
"description": "A leading provider for restaurant search and reservations"
```

Intents

```
"name": "FindRestaurants"
"description": "Find a restaurant of a particular cuisine in a city"
required_slots": ["cuisine","city"],
"optional_slots": {
   "has_live_music": "dontcare",
   "serves_alcohol": "dontcare" }
...
"name": "ReserveRestaurant"
"description": "Reserve a table at a restaurant"
"required_slots":
["restaurant_name","city","time"],
```

Slots

```
"name": "cuisine"
"description": "Cuisine of food served in the restaurant"
"is_categorical": false
"possible_values":
   ["Mexican","Chinese", "Indian",
   "American", "Italian"]
...
"name": "has_live_music"
"description": "Boolean flag indicating if the restaurant has live music"
"is_categorical": true
"possible_values": ["True","False"]
```

Dialogue

1 User: I am looking for **Chinese** food in **SFO**
System: I found 10 restaurants in **SFO**.

2 User: Any popular one? By the way, I also want to **buy a drink**.
System: There is a nice restaurant called Butterfly Restaurant.

3 User: Do they have **live music**? **Where** are they **located**?
System: They do not have live music. They are at 33 The Embarcadero.

Dialogue State Tracking

User Turn 1

intent	FindRestanurants
req_slots	N/A
slot_values	
city	**SFO**
cuisine	**Chinese**

User Turn 2

intent	FindRestanurants
req_slots	N/A
slot_values	
city	**SFO**
cuisine	**Chinese**
alcohol	**TRUE**

User Turn 3

intent	FindRestanurants
req_slots	**has_live_music, street_address**
slot_values	
city	**SFO**
cuisine	**Chinese**
alcohol	**TRUE**

Figure 1: An example dialog from Restaurant_1 service, along with its service/intent/slot descriptions and dialog state representation.

[*]Work done when Jie Cao was an intern at Amazon

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 782–796
June 6–11, 2021. ©2021 Association for Computational Linguistics

task only evaluates on a single dataset with limited variation in schema definition. It is unknown how this paradigm generalizes to other datasets and other different styles of descriptions. In this paper, we focus our investigation on the study of three aspects in schema-guided dialog state tracking: (1) schema encoding model architectures (2) supplementary training on intermediate tasks (3) various styles for schema description. To make a more general discussion on the schema-guided dialog state tracking, we perform extensive empirical studies on both SG-DST and MULTIWOZ 2.2 datasets. In summary, our contributions include:

- A comparative study on schema encoding architectures, suggesting a partial-attention encoder for good balance between inference speed and accuracy.
- An experimental study of supplementary training on schema-guided dialog state tracking, via intermediate tasks including natural language inference and question answering.
- An in-depth analysis of different schema description styles on a new suite of benchmarking datasets with variations in schema description for both SG-DST and MULTIWOZ 2.2.

2 Schema-Guided Dialog State Tracking

A classic dialog state tracker predicts a dialog state frame at each user turn given the dialog history and predefined domain ontology. As shown in Figure 1, the key difference between schema-guided dialog state tracking and the classic paradigm is the newly added natural language descriptions. In this section, we first introduce the four subtasks and schema components in schema-guided dialog state tracking, then we outline the research questions in our paper.

Subtasks. As shown in Figure 1, the dialog state for each service consists of 3 parts: *active intent*, *requested slots*, *user goals (slot values)*. Without loss of generality, for both SG-DST and MULTIWOZ 2.2 datasets, we divide their slots into categorical and non-categorical slots by following previous study on dual-strategies (Zhang et al., 2019). Thus to fill the dialog state frame for each user turn, we solve four subtasks: intent classification (*Intent*), requested slot identification (*Req*), categorical slot labeling (*Cat*), and non-categorical slot labeling (*NonCat*). All subtasks require matching the current dialog history with candidate schema descriptions for multiple times.

Schema Components. Figure 1 shows three main schema components: service, intent, slot. For each intent, the schema also describes *optional* or *required* slots for it. For each slot, there are flags indicating whether it is categorical or not. *Categorical* means there is a set of predefined candidate values (Boolean, numeric or text). For instance, *has_live_music* in Figure 1 is a categorical slot with Boolean values. *Non-categorical*, on the other hand, means the slot values are filled from the string spans in the dialog history.

New Questions. These added schema descriptions pose the following three new questions. We discuss each of them in the following sections.

Q1. How should dialogue and schema be encoded? §5

Q2. How do different supplementary trainings impact each subtask? §6

Q3. How do different description styles impact the state tracking performance? §7

3 Related Work

Our work is related to three lines of research: multi-sentence encoding, multi-domain and transferable dialog state tracking. However, our focus is on the comparative study of different encoder architectures, supplementary training, and schema description style variation. Thus we adopt existing strategies from multi-domain dialog state tracking.

Multi-Sentence Encoder Strategies. Similar to the recent study on encoders for response selection and article search tasks Humeau et al. (2019), we also conduct our comparative study on the two typical architectures *Cross-Encoder* (Bordes et al., 2014; Lowe et al., 2015) and *Dual-Encoder* (Wu et al., 2017; Yang et al., 2018). However, they only focus on sentence-level matching tasks. All subtasks in our case require sentence-level matching between dialog context and each schema, while the non-categorical slot filling task also needs to produce a sequence of token-level representation for span detection. Hence, we study multi-sentence encoding for both sentence-level and token-level tasks. Moreover, to share the schema encoding across subtasks and turns, we also introduce a simple *Fusion-Encoder* by caching schema token embeddings in §5.1, which improves efficiency without sacrificing much accuracy.

Multi-domain Dialog State Tracking. Recent research on multi-domain dialog system have been largely driven by the release of large-scale multi-domain dialog datasets, such as MultiWOZ (Budzianowski et al., 2018), M2M (Shah et al., 2018a), accompanied by studies on key issues such as in/cross-domain carry-over (Kim et al., 2019). In this paper, our goal is to understanding the design choice for schema descriptions in dialog state tracking. Thus we simply follow the in-domain cross-over strategies used in **TRADE** (Wu et al., 2019). Additionally, explicit cross-domain carryover (Naik et al., 2018) is difficult to generalize to new services and unknown carryover links. We use longer dialog history to inform the model on the dialog in the previous service. This simplified strategy does impact our model performance negatively in comparison to a well-designed dialog state tracking model on seen domains. However, it helps reduce the complexity of matching extra slot descriptions for cross-service carryover. We leave the further discussion for future work.

Transferable Dialog State Tracking. Another line of research focuses on how to build a transferable dialog system that is easily scalable to newly added intents and slots. This covers diverse topics including e.g., resolving lexical/morphological variabilities by symbolic de-lexicalization-based methods (Henderson et al., 2014; Williams et al., 2016), neural belief tracking (Mrkšić et al., 2017), generative dialog state tracking (Peng et al., 2020; Hosseini-Asl et al., 2020), modeling DST as a question answering task (Zhang et al., 2019; Lee et al., 2019; Gao et al., 2020, 2019). Our work is similar with the last class. However, we further investigate whether the DST can benefit from NLP tasks other than question answering. Furthermore, without rich description for the service/intent/slot in the schema, previous works mainly focus on simple format on question answering scenarios, such as domain-slot-type compounded names (e.g., "*restaurant-food*"), or simple question template "*What is the value for* **slot i***?*". We incorporate different description styles into a comparative discussion on §7.1.

4 Datasets

To the best of our knowledge, at the time of our study, SG-DST and MULTIWOZ 2.2 are the only two publicly available corpus for schema-guided dialog study. We choose both of them for our study. In this section, we first introduce these two representative datasets, then we discuss the generalizibility in domain diversity, function overlapping, data collecting methods.

Schema-Guided Dialog Dataset. SG-DST dataset [1] is especially designed as a test-bed for schema-guided dialog, which contains well-designed heterogeneous APIs with overlapping functionalities between services (Rastogi et al., 2019). In DSTC8 (Rastogi et al., 2020), SG-DST was introduced as the standard benchmark dataset for schema-guided dialog research. SG-DST covers 20 domains, 88 intents, 365 slots.[2] However, previous research are mainly conducted based on this single dataset and the provided single description style. In this paper, we further extended this dataset with other benchmarking description styles as shown in §7, and then we perform both homogenous and hetergenous evalution on it.

Remixed MultiWOZ 2.2 Dataset. To eliminate potential bias from the above single SG-DST dataset, we further add MULTIWOZ 2.2 (Zang et al., 2020) to our study. Among various extended versions for MultiWOZ dataset (2.0-2.3, Budzianowski et al., 2018; Eric et al., 2020; Zang et al., 2020; Han et al., 2020) , besides rectifying the annotation errors, MULTIWOZ 2.2 also introduced the schema-guided annotations, which covers 8 domains, 19 intents, 36 slots. To evaluate performance on seen/unseen services with MultiWOZ, we remix the MULTIWOZ 2.2 dataset to include as seen services dialogs related to *restaurant*, *attraction* and *train* during training, and eliminate slots from other domains/services from training split. For dev, we add two new domains *hotel* and *taxi* as unseen services. For test, we add all remaining domains as unseen, including those that have minimum overlap with seen services, such as *hospital*, *police*, *bus*. The statistics of data splits are shown in Appendix A.2. Note that this data split is different from the previous work on zero-shot MultiWOZ DST which takes a leave-one-out approach in Wu et al. (2019). By remixing the data in the way described above, we can evaluate the zero-shot performance on MultiWOZ in a way largely compatible with SG-DST.

Discussion. First, the two datasets cover diverse domains. MULTIWOZ 2.2 covers various possible dialogue scenarios ranging from requesting basic information about attractions through booking a hotel room or travelling between cities. While SG-DST covers more domains, such as 'Payments', 'Calender', 'DoctorServices' and so on.

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

[2] Please refer to the original paper for more details.

Datasets	Splits	Dialog	Domains	Services	Zero-shot Domains	Zero-shot Services	Function Overlapp	Collecting Method
SG-DST	Train	16142	16	26	-	-	Across-domain Within-domain	M2M
	Dev	2482	16	17	1	8		
	Test	4201	18	21	3	11		
MULTIWOZ 2.2	Train	9617	3	3	-	-	Across-domain	H2H
	Dev	2455	5	5	2	2		
	Test	2969	8	8	5	5		

Table 1: Summary of characteristics of SG-DST MULTIWOZ 2.2 datasets, in domain diversity, function overlap, data collecting methods

Second, they include different levels of overlapping functionalities. SG-DST allows frequent function overlapping between multiple services, within the same domain (e.g. BookOneWayTicket v.s. BookRoundTripTicket), or across different domains (BusTicket v.s. TrainTicket). However, the overlapping in MULTIWOZ 2.2 only exists across different domains, e.g., 'destination', 'leaveat' slots for Taxi and Bus services, 'pricerange', 'bookday' for Restaurant and Hotel services.

Third, they are collected by two different approaches which are commonly used in dialog collecting. SG-DST is firstly collected by machine-to-machine self-play (M2M, Shah et al., 2018b) with dialog flows as seeds, then paraphrased by crowd-workers. While MULTIWOZ 2.2 are human-to-human dialogs (H2H, Kelley, 1984), which are collected with the Wizard-of-Oz approach.

We summarize the above discussion in Table 1. We believe that results derived from these two representative datasets can guide future research in schema guided dialog.

5 Dialog & Schema Representation and Inference (Q1)

In this section, we focus on the model architecture for matching dialog history with schema descriptions using pretrained BERT (Devlin et al., 2019) [3]. To support four subtasks, we first extend *Dual-Encoder* and *Cross-Encoder* to support both sentence-level matching and token-level prediction. Then we propose an additional *Fusion-Encoder* strategy to get faster inference without sacrificing much accuracy. We summarize different architectures in Figure 2. Then we show the classification head and results for each subtask.

[3]We use BERT-base-cased for all main experiments. Other pretrained language models can be easily adapted to our study

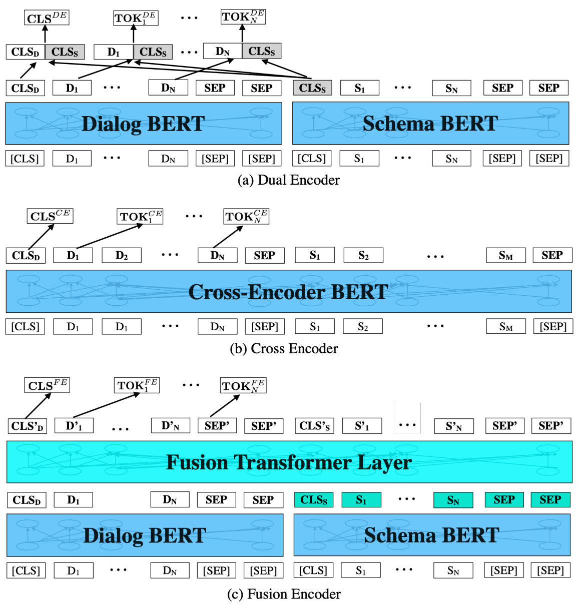


Figure 2: Dual-Encoder, Cross-Encoder and Fusion Encoder, shaded block will be cached during training

5.1 Encoder Architectures

Dual-Encoder. It consists of two separate BERTs to encode dialog history and schema description respectively, as Figure 2 (a). We follow the setting in the official baseline provided by DSTC8 Track4 (Rastogi et al., 2020). We first use a fixed BERT to encode the schema description once and cached the encoded schema $\mathbf{CLS}_S$. Then for sentence-level representation, we concatenate dialog history representation $\mathbf{CLS}_D$ and candidate schema representation $\mathbf{CLS}_S$ as the whole sentence-level representation for the pair, denoted as $\mathbf{CLS}^{DE}$. For token-level representation, we concatenate the candidate schema $\mathbf{CLS}_S$ with each token embedding in the dialog history, denoted as $\mathbf{TOK}^{DE}$.[4] Because the candidate schema embeddings are encoded independently from the di-

[4]A schema-aware dialog token embedding can also be computed by attention or other method for span-based detection tasks (Humeau et al., 2019; Noroozi et al., 2020)

alog context, they can be pre-computed once and cached for fast inference.

Cross-Encoder. Another popular architecture as Figure 2 (b) is *Cross-Encoder*, which concatenates the dialog and schema as a single input, and encodes jointly with a single self-attentive encoder spanning over the two segments. When using BERT to encode the concatenated sentence pair, it performs full (cross) self-attention in every transformer layers, thus offer rich interaction between the dialog and schema. BERT naturally produces a summarized representation with [CLS] embedding $\mathbf{CLS}^{CE}$ and each schema-attended dialog token embeddings $\mathbf{TOK}^{CE}$. Since the dialog and schema encoding always depend on each other, it requires recomputing dialog and schema encoding for multiple times, thus much slower in inference.

Fusion-Encoder. In Figure 2 (c), similar to *Dual-Encoder*, *Fusion-Encoder* also encodes the schema independently with a fixed BERT and finetuning another BERT for dialog encoding. However, instead of caching a single [CLS] vector for schema representation, it caches **all token representation** for the schema including the [CLS] token. What's more, to integrate the sequences dialog token representation with schema token representation, an extra stack of transformer layers are added on top to allow token-level fusion via self-attention, similar to *Cross-Encoder*. The top transformer layers will produce embeddings for each token $\mathbf{TOK}^{FE}$ including a schema-attended $\mathbf{CLS}^{FE}$ of the input [CLS] from the dialog history. With cached schema token-level representations, it can efficiently produce schema-aware sentence- and token-level representation for each dialog-schema pairs.

5.2 Model Overview

All the above 3 encoders will produce both sentence- and token-level representations for a given sentence pair. In this section, we abstract them as two representations **CLS** and **TOK**, and present the universal classification heads to make decisions for each subtask.

Active Intent. To decide the intent for current dialog turn, we match current dialog history D with each intent descriptions $I_0...I_k$. For each dialog-intent pair (D, I_k), we project the final sentence-level **CLS** representation to a single number $P_{I_k}^{active}$ with a linear layer follows a sigmoid function. We predict *"NONE"* if the $P_{I_k}^{active}$ of all intents are less than a threshold 0.5, which means no intent is active. Otherwise, we predict the intent with largest $P_{I_k}^{active}$. We predict the intent for each turn independently without considering the prediction on previous turns.

Requested Slot. As in Figure 1, mulitple requested slots can exist in a single turn. We use the same strategy as in active intent prediction to predict a number P_{req}^{active}. However, to support the multiple requested slots prediction. We predict all the requested slots with $P_{req}^{active} > 0.5$.

Categorical Slot. Categorical slots have a set of candidate values. We cannot predict unseen values via n-way classification. Instead, we do binary classification on each candidate value. Besides, rather than directly matching with values, we also need to check that whether the corresponding slot has been activated. For *Cross-Encoder* and *Fusion-Encoder*, we use typical two-stage state tracking to incrementally build the state: **Step** 1. Using **CLS** to predict the slot status as *none*, *dontcare* or *active*. When the status is *active*, we use the predicted slot value; Otherwise, it will be assigned to *dontcare* meaning no user preference for this slot, or *none* meaning no value update for the slot in current turn; **Step** 2. If Step 1 is *active*, we match the dialog history with each value and select the most related value by ranking. We train on cross entropy loss. Two-stage strategy is efficient for *Dual-Encoder* and *Fusion-Encoder*, where cached schema can be reused, and get efficiently ranked globally in a single batch. However, it is not scalable for *Cross-Encoder*, especially for large number of candidate values in MultiWOZ dataset. Hence, during training, we only use a binary cross-entropy for each single value and postpone the ranking only to the inference time.

Noncategorical Slot. The slot status prediction for noncategorical slot use the same two-stage strategy. Besides that, we use the token representation of dialog history **TOK** to compute two softmax scores f_{start}^{i} and f_{end}^{i} for each token i, to represent the score of predicting the token as start and end position respectively. Finally, we find the valid span with maximum sum of the start and end scores.

5.3 Experiments on Encoder Comparison

To fairly compare all three models, we follow the same schema input setting as in Table 2. We trained separate models for SG-DST and the remixed MultiWOZ datasets for all the experiments in our pa-

Intent	service description, intent description
Req	service description, slot description
Cat	slot description, cat value
NonCat	service description, slot description

Table 2: Schema description input used for different tasks to compare *Dual-Encoder*, *Cross-Encoder*, and *Fusion-Encoder*. In the appendix A.3, we also studies other compositions of description input. We found that service description will not help for *Intent*, *Req* and *Cat* tasks, while the impact on *NonCat* task also varies from SG-DST and MULTIWOZ 2.2 dataset.

	SG-DST					MULTIWOZ 2.2		
Method/Task	Acc	F1	Joint Acc			Joint Acc		
	Intent	Req	Cat	NonCat	All	Cat	NonCat	All
Seen Services								
Dual-Encoder	94.51	99.62	87.92	47.77	43.20	79.20	79.34	65.64
Fusion-Encoder	94.90	**99.69**	88.94	48.78	58.52	81.37	80.58	67.43
Cross-Encoder	**95.55**	99.59	**93.68**	**91.85**	**87.58**	**85.99**	**81.02**	**71.93**
Unseen Services								
Dual-Encoder	89.73	95.20	42.44	31.62	19.51	56.92	50.82	31.83
Fusion-Encoder	90.47	95.95	48.79	35.91	22.85	57.01	52.23	33.64
Cross-Encoder	**93.84**	**98.26**	**71.55**	**74.13**	**54.54**	**59.85**	**59.62**	**38.46**

Table 3: Test set results on SG-DST and MULTIWOZ 2.2. The *Dual-Encoder* model is a re-implementation of official DSTC8 baseline from Rastogi et al. (2019). Other models are trained with the architecture described in our paper.

pers[5]. Because there are very few intent and requested slots in MULTIWOZ 2.2 dataset, we ignore the intent and requested slots tasks for MULTIWOZ 2.2 in our paper.

Results. As shown in Table 3, *Cross-Encoder* performs the best over all subtasks. Our *Fusion-Encoder* with partial attention outperforms the *Dual-Encoder* by a large margin, epsecially on categorical and noncategorical slots predictions. Additionally, on seen services, we found that *Dual-Encoder* and *Fusion-Encoder* can perform as good as *Cross-Encoder* on *Intent* and *Req* tasks. However, they cannot generalize well on unseen services as *Cross-Encoder*.

Inference Speed. To test the inference speed, we conduct all the experiments with a maximum affordable batch size to fully exploit 2 V100 GPUs (with 16GB GPU RAM each). During training, we log the inference time of each evaluation on dev set. Both *Dual-Encoder* and *Fusion-Encoder* can do joint inference across 4 subtasks to obtain an integral dialog state for a dialog turn example. *Dual-Encoder* achieves the highest inference speed of **603.35** examples per GPU second, because the encoding for dialog and schema are fully separated. A dialog only needed to be encoded for once during the inference of a dialog state example while the schema are precomputed once. However, for *Cross-Encoder*, to predict a dialog state for a single turn, it need to encode more than 300 sentence pairs in a batch, thus only processes **4.75** examples per GPU second. *Fusion-Encoder* performs one time encoding on dialog history, but it needs to jointly encode the same amount of dialog-schema pair ws *Cross-Encoder*, instead, however, with a two-layer transformer encoder. Overall it achieves **10.54** examples per GPU second, which is **2.2x** faster than *Cross-Encoder*. With regarding to the accuracy in Table 3, *Fusion-Encoder* performs much better than *Dual-Encoder*, especially on unseen services.

6 Supplementary Training (Q2)

Besides the pretrain-fintune framework used in §5, Phang et al. (2018) propose to add a supplementary training phase on an intermediate task after the pretraining, but before finetuning on target task. It shows significant improvement on the target tasks. Moreover, large amount pretrained and finetuned transformer-based models are publicly accessible, and well-organized in model hubs for sharing, training and testing[6]. Given the new task of schema-guided dialog state tracking, in this section, we study our four subtasks with different intermediate tasks for supplementary training.

6.1 Intermediate Tasks

As described in § 5.2, all our 4 subtasks take a pair of dialog and schema description as input, and predict with the summerized sentence-pair **CLS** representation. While *NonCat* also requires span-based detection such as question answering. Hence, they share the similar problem structure with the following sentence-pair encoding tasks.

Natural Language Inference. Given a hypothesis/premise sentence pair, natural language inference is a task to determine whether a hypothesis is entailed, contradicted or neutral given that premise.

Question Answering. Given a passage/question pairs, the task is to extract the span-based answer in the passage.

Hence, when finetuning BERT on our subtaks, instead of directly using the originally pretrained BERT, we use the BERT finetuned on the above

[5]Appendix A.1 shows the detailed experiment setup

[6]e.g., Huggingface(https://huggingface.co/models) and ParlAL(https://parl.ai/docs/zoo.html), etc.

	SG-DST												MultiWOZ 2.2					
	intent			req			cat			noncat			cat			noncat		
	all	seen	unseen	all	seen	unseen	all	seen	unseen	all	seen	unseen	all	seen	unseen	all	seen	unseen
Δ_{SNLI}	**+0.51**	+0.02	**+0.68**	-0.19	+0.38	-0.38	-1.63	-2.87	-1.23	-4.7	-0.1	-6.25	+2.05	+0.6	–0.7	**+3.64**	+1.05	**+4.84**
Δ_{SQuAD}	-1.81	-0.17	-1.32	-0.25	-0.01	-0.33	-2.87	-3.02	-5.17	**+1.99**	-1.79	**+3.25**	+0.04	-0.71	+0.41	**+1.93**	-2.21	**+4.27**

Table 4: Relative performance improvement of different supplementary training on SG-DST and MultiWOZ 2.2 dataset

two tasks for further finetuning. Due to better performance of *Cross-Encoder* in §5, we directly use the finetuned *Cross-Encoder* version of BERT models on SNLI and SQuAD2.0 dataset from Huggingface model hub. We add extra speaker tokens [user:] and [system:] into the vocabulary for encoding the multi-turn dialog histories.

6.2 Results on Supplementary Training

Table 4 shows the performances gain when finetuning 4 subtasks based on models with the above SNLI and SQuAD2.0 supplementary training.

We mainly find that SNLI helps on *Intent* task, SQuAD2 mainly helps on *NonCat* task, while neither of them helps much on *Cat* task. Recently, Namazifar et al. (2020) also found that when modeling dialog understanding as question answering task, it can benefit from a supplementary training on SQuAD2 dataset, especially on few-shot scenarios, which is a similar findings as our *NonCat* task. Result difference on *Req* task is minor, because it is a relatively easy task, adding any supplementary training did n't help much. Moreover, for *Cat* task, the sequence 2 of the input pair is the slot description with a categorical slot value, thus the meaning overlapping between the full dialog history and the slot/value is much smaller than SNLI tasks. On the other side, **CLS** token in SQuAD BERT is finetuned for null predictions via start and end token classifers, which is different from the the single CLS classifer in *Cat* task.

7 Impact of Description Styles (Q3)

Previous work on schema-guided dialog (Rastogi et al., 2020) are only based on the provided descriptions in SG-DST dataset. Recent work on modeling dialog state tracking as reading comprehension (Gao et al., 2019) only formulate the descriptions as simple question format with existing intent/slot names, it is unknown how it performs when compared to other description styles. Moreover, they only conduct homogeneous evaluation where training and test data share the same description style. In this section, We also investigate how a model trained on one description style will perform on other different styles, especially in a scenario where chat-bot developers may design their own descriptions. We first introduce different styles of descriptions in our study, and then we train models on each description style and evaluate on tests with corresponding homogeneous and heterogeneous styles of descriptions. Given the best performance of *Cross-Encoder* shown in the previous section and its popularity in DSTC8 challenges, we adopt it as our model architecture in this section.

7.1 Benchmarking Styles

For each intent/slot, we describe their functionalities by the following different descriptions styles:

***Identifer*.** This is the least informative case of name-based description: we only use meaningless intent/slot identifiers, e.g. Intent_1, Slot_2. It means we don't use description from any schema component. We want to investigate how a simple identifier-based description performs in schema-guided dialog modeling, and the performance lower-bound on transferring to unseen services.

***NameOnly*.** Using the original intent/slot names in SG-DST and MultiWOZ 2.2 dataset as descriptions, to show whether name is enough for schema-guided dialog modeling.

***Q-Name*.** This is corresponding to previous work by Gao et al. (2019). For each intent/slot, it generate a question to inquiry about the intent and slot value of the dialog. For each slot, it simply follows the template '*What is the value for* **slot i**?'. Besides that, our work also extend the intent description by following the template "*Is the user intending to* **intent j** ".

***Orig*.** The original descriptions in SG-DST and MultiWOZ 2.2 dataset.

***Q-Orig*.** Different from the *Q-Name*, firstly it is based on the original descriptions; secondly, rather than always use the "what is" template to inquiry the intent/slot value, We add "what", "which", "how many" or "when" depending on the entity type required for the slot. Same as *Q-Name*, we just add prefixes as "Is the user intending to..." in front of the original description. In a sum, this description is just adding question format to original description. The motivation of this description is to see whether the question format is helpful or not for schema-guided dialog modeling.

To test the model robustness, we also create two paraphrased versions *Name-Para* and *Orig-Para* for *NameOnly* and *Orig* respectively. We first use nematus (Sennrich et al., 2017) to automatically paraphrase the description with back translation, from English to Chinese and then translate back, then we manually check the paraphrase to retain the main meaning. Appendix A.5.1 shows examples for different styles of schema descriptions.

7.2 Results on Description Styles

Unlike the composition used in Table 2, we don't use the service description to avoid its impact. For each style, we train separate models on 4 subtasks, then we evaluate them on different target styles. First, Table 5 summarizes the performance for homogeneous evaluation, while Table 6 shows how the question style description can benefit from SQuAD2 finetuning. Then we also conduct heterogeneous evaluation on the other styles[7] as shown in Table 7.

Style\Task	SG-DST				MULTIWOZ 2.2	
	Intent	Req	Cat	NonCat	Cat	NonCat
Identifer	61.16	91.48	62.47	30.19	34.25	52.28
NameOnly	**94.24**	98.84	74.01	75.63	53.72	56.18
Q-Name	93.31	**98.86**	74.36	74.86	**54.19**	56.17
Orig	93.01	98.55	74.51	75.76	52.19	57.20
Q-Orig	93.42	98.51	**76.64**	**76.60**	53.61	**57.80**

Table 5: Homogeneous evaluation results of different description style on SG-DST dataset and MULTIWOZ 2.2 datasets. The middle horizontal line separate the two name-based descriptions and two rich descriptions in our settings. All numbers in the table are mixed performance including both seen and unseen services.

7.2.1 Homogeneous Evaluation

Is name-based description enough? As shown in Table 5, *Identifer* is the worst case of using name description, its extremely bad performance indicates name-based description can be very unstable. However, we found that simple meaningful name-based description actually can perform the best in *Intent* and *Req* task, and they perform worse on *Cat* and *NonCat* tasks comparing to the bottom two *rich* descriptions. [8] After careful analysis on the intents in SG-DST datasets, we found that most services only contains two kinds of intents, an information retrieval intent with a name prefix "Find-", "Get-", "Search-"; another transaction intent like "Add-", "Reserve-" or "Buy-". Interestingly, we found that all the intent names in the original schema-guided dataset strictly follows an action-object template with a composition of words without abbreviation, such as "FindEvents", "BuyEventTickets". This simple name template is good enough to describe the core functionality of an intent in SG-DST dataset. [9] Additionally, *Req* is a relaitively simper task, requesting information are related to specifial attributes, such as "has_live_music", "has_wifi", where keywords cooccured in the slot name and in the user utterance, hence rich explanation cannot help further. On the other side, *rich* descriptions are more necessary for *Cat* and *NonCat* task. Because in many cases, slot names are too simple to represent the functionalities behind it, for example, slot name "passengers" cannot fully represent the meaning "number of passengers in the ticket booking".

Does question format help? As shown in Table 5, when comparing row *Q-Orig* v.s. *Orig*, we found extra question format can improve the performance on *Cat* and *NonCat* task on both SG-DST and MULTIWOZ 2.2 datasets, but not for *Intent* and *Req* tasks. We believe that question format helps the model to focus more on specific entities in the dialog history. However, when adding a simple question pattern to *NameOnly*, comparing row *Q-Name* and *NameOnly*, there is no consistent improvement on both of the two datasets. Further more, we are curious about whether BERT finetuned on SQuAD2 (SQuAD2-BERT) can further help on the question format. Because *NonCat* are similar with span-based question answering, we focus on *NonCat* here. Table 6 shows that, after applying the supplementary training on SQuAD2 (§6), almost all models get improved on unseen splits however slightly dropped on seen services. Moreover, comparing to *Q-Name*, *Q-*

[7]We don't consider the meaningless *Identifer* style due to its bad performance

[8]Only exception happens in *Cat* on MULTIWOZ 2.2. When creating MULTIWOZ 2.2 (Zang et al., 2020), the slots with less than 50 different slot values are classified as categorical slots, which leads to inconsistencies. We put detailed discuss about MULTIWOZ 2.2 in the supplementary material

[9]This action-object template has also been found efficient for open domain intent induction task(e.g., Vedula et al., 2020, OPINE).

Style/Dataset	SG-DST			MULTIWOZ 2.2		
	all	seen	unseen	all	seen	unseen
Orig	+1.99	-1.79	**+3.25**	+1.93	-2.21	**+4.27**
Q-Orig	+6.13	-2.01	**+8.84**	+1.06	-1.28	**+3.06**
NameOnly	-0.45	-1.49	-0.11	+1.75	+0.58	**+1.77**
Q-Name	+0.05	-2.98	**+1.04**	-0.04	-0.32	**+1.25**

Table 6: Performance changes when using BERT finetuned on SQuAD2 dataset to further finetuning on our *NonCat* task.

Orig is more similar to the natural questions in the SQuAD2, we obverse that *Q-Orig* gains more than *Q-Name* from pretrained model on SQuAD2.

7.2.2 Heterogeneous

In this subsection, we first simulate a scenario when there is no recommended description style for the future unseen services. Hence, unseen services can follow any description style in our case. We average the evaluation performance on three other descriptions and summarized in Table 7. The Δ column shows the performance change compared to the homogeneous performance. It is not surprising that almost all models perform worse on heterogeneous styles than on homogeneous styles due to different distribution between training and evaluation. The bold number shows the best average performance on heterogeneous evaluation for each subtask. The trends are similar with the analysis in homogeneous evaluation 7.2.1, the name-based descriptions perform better than other rich descriptions on intent classification tasks. While on other tasks, the *Orig* description performs more robust, especially on *NonCat* task.

Furthermore, we consider another scenario where fixed description convention such as *NameOnly* and *Orig* are suggested to developers, they must obey the basic style convention but still can freely use their own words, such as abbreviation, synonyms, adding extra modifiers. We train each model on *NameOnly* and *Orig*, then evaluate on the corresponding paraphrased version respectively. In the last two rows of Table 7, the column 'para' shows performance on paraphrased schema, while Δ shows the performance change compared to the homogeneous evaluation. *Orig* still performs more robust than *NameOnly* when schema descriptions get paraphrased on unseen services.

Style\Task	SG-DST							
	Intent(Acc)		Req(F1)		Cat(Joint Acc)		NonCat(Joint Acc)	
	mean	Δ	mean	Δ	mean	Δ	mean	Δ
NameOnly	82.47	-11.47	96.92	-1.64	61.37	-5.54	56.53	-14.68
Q-Name	**93.27**	+0.58	**97.88**	-0.76	68.55	+2.63	62.92	-6.30
Orig	79.47	-12.70	97.42	-0.74	**68.58**	-0.3	**66.72**	-3.11
Q-Orig	84.57	-8.24	96.70	-1.45	68.40	-2.89	56.17	-15.00
	para	Δ	para	Δ	para	Δ	para	Δ
NameOnly	**92.22**	-1.74	97.69	-0.87	67.39	-0.7	67.17	-4.04
Orig	91.54	**-0.63**	**98.42**	**+0.26**	**71.74**	**+2.86**	**67.68**	**-2.16**

Table 7: Results on unseen service with heterogeneous description styles on SG-DST dataset. More results and qualitative analysis are in the appendix A.5

8 Conclusion

In this paper, we studied three questions on schema-guided dialog state tracking: encoder architectures, impact of supplementary training, and effective schema description styles. The main findings are as follows:

By caching the token embedding instead of the single CLS embedding, a simple partial-attention *Fusion-Encoder* can achieve much better performance than *Dual-Encoder*, while still infers two times faster than *Cross-Encoder*. We quantified the gain via supplementary training on two intermediate tasks. By carefully choosing representative description styles according to recent works, we are the first of doing both homogeneous/heterogeneous evaluations for different description style in schema-guided dialog. The results show that simple name-based description performs well on *Intent* and *Req* tasks, while *NonCat* tasks benefits from richer styles of descriptions. All tasks suffer from inconsistencies in description style between training and test, though to varying degrees.

Our study are mainly conducted on two datasets: SG-DST and MULTIWOZ 2.2, while the speed-accuracy balance of encoder architectures and the findings in supplementary training are expected to be dataset-agnostic, because they depend more on the nature of the subtasks than the datasets. Based on our proposed benchmarking descriptions suite, the homogeneous and heterogeneous evaluation has shed the light on the robustness of cross-style schema-guided dialog modeling, we believe our study will provide useful insights for future research.

Acknowledgments

The authors wish to thank the anonymous reviewers and members of the Amazon LEX team for their valuable feedback.

References

Daniel G Bobrow, Ronald M Kaplan, Martin Kay, Donald A Norman, Henry Thompson, and Terry Winograd. 1977. Gus, a frame-driven dialog system. *Artificial intelligence*, 8(2):155–173.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 165–180. Springer.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Daniel Duckworth, Semih Yavuz, Ben Goodrich, Amit Dubey, Andy Cedilnik, and Kyu-Young Kim. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. *arXiv preprint arXiv:1909.05358*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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

Shuyang Gao, Sanchit Agarwal, Tagyoung Chung, Di Jin, and Dilek Hakkani-Tur. 2020. From machine reading comprehension to dialogue state tracking: Bridging the gap. *arXiv preprint arXiv:2004.05827*.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, Dilek Hakkani-Tur, and Amazon Alexa AI. 2019. Dialog state tracking: A neural reading comprehension approach. In *20th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 264.

Ting Han, Ximing Liu, Ryuichi Takanobu, Yixin Lian, Chongxuan Huang, Wei Peng, and Minlie Huang. 2020. Multiwoz 2.3: A multi-domain task-oriented dataset enhanced with annotation corrections and co-reference annotation. *arXiv e-prints*, pages arXiv–2010.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *ICLR*.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2019. Efficient dialogue state tracking by selectively overwriting memory. *arXiv preprint arXiv:1911.03906*.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. Sumbt: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483.

Ryan Lowe, Nissan Pow, Iulian Vlad Serban, and Joelle Pineau. 2015. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788.

Chetan Naik, Arpit Gupta, Hancheng Ge, Mathias Lambert, and Ruhi Sarikaya. 2018. Contextual slot carry-over for disparate schemas. *Proc. Interspeech 2018*, pages 596–600.

Mahdi Namazifar, Alexandros Papangelis, Gokhan Tur, and Dilek Hakkani-Tür. 2020. Language model is all you need: Natural language understanding as question answering. *arXiv preprint arXiv:2011.03023*.

Vahid Noroozi, Yang Zhang, Evelina Bakhturina, and Tomasz Kornuta. 2020. A fast and robust bert-based dialogue state tracker for schema-guided dialogue dataset. *Workshop on Conversational Systems Towards Mainstream Adoption at KDD 2020*.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model. *arXiv*, pages arXiv–2005.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *AAAI 2019*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Schema-guided dialogue state tracking task at dstc8. *Workshop on DSTC8, AAAI 2020*.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nădejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.

Pararth Shah, Dilek Hakkani-Tür, Bing Liu, and Gokhan Tür. 2018a. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51, New Orleans - Louisiana. Association for Computational Linguistics.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018b. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Nikhita Vedula, Nedim Lipka, Pranav Maneriker, and Srinivasan Parthasarathy. 2020. Open intent extraction from natural language interactions. In *Proceedings of The Web Conference 2020*, pages 2009–2020.

Jason D Williams, Antoine Raux, and Matthew Henderson. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*, 7(3):4–33.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819.

Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. 2017. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505.

Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W Bruce Croft, Jun Huang, and Haiqing Chen. 2018. Response ranking with deep matching networks and external knowledge in information-seeking conversation systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 245–254.

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, Online. Association for Computational Linguistics.

Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv*, pages arXiv–1910.

A Appendices

A.1 Experiment Setup

All models are based on BERT-base-cased model with 2 V100 GPUs (with 16GB GPU RAM each). We train each models for maximum 10 epoch, by using AdamW to schedule the learning rate with a warm-up portion of 0.1. During training, we evaluate checkpoints per 3000 steps on dev splits, and select the model with best performance on dev split on all seen and unseen services. In our experiments, our model achieves the best performance on around 2-4 epochs on *Intent*, *Req*. and *Cat*, while *NonCat* needs 5-8 epochs to get the best performance. For all subtasks, as we model all of them as sentence pair encoding during training, we use batch size as 16 for each GPU, and gradient accumulate for 8 steps, in total 256 batch size on 2 GPUs.

A.2 Statistic on MultiWOZ 2.2 Remix

To evaluate performance on seen/unseen services with MultiWOZ, we remix the MULTIWOZ 2.2 dataset to include as seen services dialogs related to *restaurant*, *attraction* and *train* during training, and eliminate slots from other domains/services from training split. For dev, we add two new domains *hotel* and *taxi* as unseen services. For test, we add all remaining domains as unseen, including those that have minimum overlap with seen services, such as *hospital*, *police*, *bus*. The statistics are as shown in Table 8

Domain	#dialogs/#turns					
	train		dev		test	
restaurant	3900	37953	458	6979	451	7104
attraction	2716	28632	405	6198	400	6290
train	3001	29646	481	5897	491	6150
hotel	0	0	737	8509	718	7911
taxi	0	0	374	2692	364	2659
hospital	0	0	0	0	287	766
police	0	0	0	0	252	475
bus	0	0	0	0	6	132

Table 8: The total number of dialogs and turns related to each domain in train, dev and test split of MultiWOZ

A.3 Composition of Descriptions

A.3.1 Composition Settings

For each subtask, the key description element must be included, e.g., intent description for intent task, and value for categorical slot tasks. To show how each component helps schema-guided dialog state tracking, we incrementally add richer schema component one by one.

ID. This is the least informative case: we only use meaningless intent/slot identifiers, e.g. Intent_4, Slot_2. It means we don't use description from any schema component. We want to investigate how a simple identifier-based description performs in schema-guided dialog modeling, and the performance lower-bound on transferring to unseen services.

I/S Desc. Only using the original intent/slot description of intent/slot in SG-DST and MULTIWOZ 2.2 dataset for corresponding tasks.

Service + I/S Desc. Adding a service description to the above original description. Service description summarize the functionalities of the whole service, hence may offer extra background information for intent and slots. For categorical slot value detection, we simply add the value after each of the above composition.

A.3.2 Results on Description Compositions

Table 9 shows the results of using different description compositions. First, there are consistent findings across datasets and subtasks: (1) using meaningless identifier as intent/slot description shows the worse performance on all tasks of both datasets, and can not generalize well to unseen services. (2) using intent/slot descriptions can largely boost the performance, especially on unseen services.

Model\Task	SG-DST				MultiWOZ	
	Intent	Req	Cat	NonCat	Cat	NonCat
Seen Service						
Identifer	92.76	99.70	87.86	88.38	58.46	77.29
I/S Desc	**95.35**	**99.74**	92.10	**93.52**	**85.84**	**83.67**
Service + I/S Desc	95.28	**99.74**	**93.19**	92.34	85.07	80.56
Unseen Service						
Identifier	50.63	88.74	54.34	10.77	53.05	56.18
I/S Desc	**92.17**	**98.16**	**68.88**	69.84	56.49	**61.39**
Service + I/S Desc	86.95	97.99	67.08	**71.30**	**60.58**	59.63

Table 9: Models using different composition of schema, results on test set of SG-DST and our remixed MULTIWOZ 2.2

However, the impact of service description varies by tasks. For example, it largely hurts performance on intent classification task, but does not impact requested slot and categorical slot tasks. According to manual analysis of SG-DST and MULTIWOZ 2.2 dataset, we found that service description consists of the main functions of the service, especially the meaning of the supported intents. Hence, using service description for intent causes confusion between the intent description information and other supported intents. Moreover, in categorical slot value prediction task, the most important information is the slot description and value. When adding extra information from service description, it improves marginally on seen service while not generalizing well on unseen services, which indicates the model learns artifacts that are not general useful for unseen services.

Finally, on non-categorical slot tasks, the impact of service description may also varies on datasets. On SG-DST, there are 16 domains and more than 30 services, the rich background context from service description contains both domain and service-specific information, which seems to help both seen and unseen services. However, on MULTIWOZ 2.2, it hurts the performance on seen service *restaurant* the most, while improving the performance on the unseen service *hotel* by 4 points. In this case, it works like a regularizer rather than a definitive clues. Because in MULTIWOZ 2.2, there are only 8 domains, and one service per domain, thus service descriptions just contain domain related information without much extra information, it will not help the model to detect the span for the slot.

A.4 More Results of Supplementary Training

Table 10 shows the detailed performance when using different intermediate tasks as supplementary training. For SNLI tasks, as the pretrained model is uncased model (textattack/ bert-base-uncased-snli),

		sgd												multiwoz					
		intent			req			cat			noncat			cat			noncat		
snli	uncased	93.31	95.4	92.62	98.62	99.34	98.37	75.66	93.39	69.98	80.38	90.93	76.87	51.77	84.93	59.40	56.47	82.39	61.62
	snli	93.82	95.42	93.3	98.43	99.72	97.99	74.03	90.52	68.75	75.68	90.83	70.62	53.82	85.53	58.70	60.11	83.44	66.46
	$\triangle_{\text{SNLI}}$	**+0.51**	+0.02	**+0.68**	-0.19	+0.38	-0.38	-1.63	-2.87	-1.23	-4.7	-0.1	-6.25	+2.05	+0.6	–0.7	**3.64**	+1.05	**+4.84**
squad	cased	93.01	95.51	92.2	98.59	99.59	98.26	74.51	92.1	71.23	75.76	93.52	69.84	52.19	85.74	56.49	57.2	83.67	61.39
	squad	91.2	95.34	90.88	98.34	99.58	97.93	71.64	89.08	66.06	77.75	91.73	73.09	52.23	85.03	56.90	59.13	81.46	65.66
	$\triangle_{\text{SQuAD}}$	-1.81	-0.17	-1.32	-0.25	-0.01	-0.33	-2.87	-3.02	-5.17	**1.99**	-1.79	**3.25**	+0.04	-0.71	+0.41	**1.93**	-2.21	**+4.27**

Table 10: Results of different supplementary training on SG-DST and MULTIWOZ 2.2 dataset

style	Intent Description	Slot Description
Identifer	intent_1	slot_4
NameOnly	CheckBalance	account_type
Q-Name	Is the user intending to CheckBalance?	What is the value of acctount_type ?
Orig	Check the amount of money in a user's bank account	The account type of the user
Q-Orig	Does the user want to check the amount of money in the bank account ?	What is the account type of the user ?
Name-Para	CheckAccountBalance	user_account_type
Orig-Para	Check the balance of the user's bank account	Type of the user account

Table 11: Different extensions of schema descriptions

hence, we first train different models with BERT-base-uncased, then compare the performance with SNLI pretrained model. For SQuAD2, we use deepset/bert-base-cased-SQuAD2 model, hence, we compare it all cased model. To fairly compare with our original *Cross-Encoder*, we add extra speaker tokens [user:] and [system:] for encoding the multi-turn dialog histories.

A.5 Homogeneous and Hetergenuous Evaluation on Different Styles

A.5.1 Examples for Different Description Styles

Table 11 shows examples for different styles of schema descriptions.

A.5.2 More details on SQuAD2 Results on Different Styles

For homogeneous evaluation, Table 12 shows the detailed performance when we apply SQuAD2-finetuned BERT on our models.

A.5.3 More Results On Homogeneous and Heterogeneous Evalution

We list the detailed results for our evaluation across different styles. We use *italic* to show the homogeneous evaluation, where the results are shown in the diagonal of each table, and we underline the best homogeneous results in the diagonal. We use **bold** to show the best heterogeneous performance and the best performance gap in the last two columns

Intent. The results on SG-DST dataset are shown in Table 13. Because there are very few intents in MULTIWOZ 2.2 dataset, we don't conduct intent classification on MULTIWOZ 2.2. All performance get dropped when evaluating on heterogeneous descriptions styles. For both heterogeneous and homogeneous evaluation, adding rich description on intent classification tasks seems not bring much benefits than simply using the named-based description. As the discussion in §7.2.1, we believe the name template is good enough to describe the core functionality of an intent in SG-DST dataset.

Style/Dataset	SG-DST			MULTIWOZ 2.2		
	all	seen	unseen	all	seen	unseen
Orig	75.76	93.52	69.84	57.2	83.67	61.39
	77.75	91.73	73.09	59.13	81.46	65.66
	+1.99	-1.79	**+3.25**	+1.93	-2.21	**+4.27**
Q-Orig	76.60	92.86	71.18	57.80	82.45	62.45
	82.73	90.85	80.02	58.86	81.17	65.51
	+6.13	-2.01	**+8.84**	+1.06	-1.28	**+3.06**
NameOnly	75.63	88.90	71.21	56.18	81.68	61.30
	75.18	87.41	71.10	57.93	82.26	63.07
	-0.45	-1.49	**–0.11**	+1.75	+0.58	**+1.77**
Q-Name	74.86	91.78	69.22	56.17	81.19	60.47
	74.91	88.8	70.26	56.13	80.87	61.72
	+0.05	-2.98	**+1.04**	-0.04	-0.32	**+1.25**

Table 12: Results on different description style on SG-DST and MULTIWOZ 2.2 dataset, when performing SQuAD2 supplementary training

Requested Slot. Table 14 shows the results on SG-DST dataset for the requested slots subtask. We ignore the requested slots in MULTIWOZ 2.2 dataset due to its sparsity. Overall, the requested slot subtask are relatively easy, performances on heterogeneous styles still drops but not much. For both heterogeneous and homogeneous evaluation, the performance are not sensible to rich description.

Categorical Slot. The results on SG-DST and MULTIWOZ 2.2 dataset are shown in Table 15.

Style	*NameOnly*	*Q-Name*	*Orig*	*Q-Orig*	mean	Δ
NameOnly	*93.94*	78.27	93.18	75.95	82.47	-11.47
Q-Name	93.18	*92.69*	93.26	93.36	**93.27**	**+0.58**
Orig	81.57	66.42	*92.17*	90.43	79.47	-12.70
Q-Orig	81.48	79.04	93.19	*92.81*	84.57	-8.24

Table 13: Accuracy of intent classification subtask with different description styles on unseen services. Train the model on SG-DST dataset for each description in each row, then evaluating on 4 different descriptions styles. The mean are average performance of the remaining 3 descriptions styles. The Δ means the performance gap between the mean and the homogeneous performance

Style	*NameOnly*	*Q-Name*	*Orig*	*Q-Orig*	mean	Δ
NameOnly	*98.56*	96.01	97.2	97.54	96.92	-1.64
Q-Name	98.37	*98.64*	97.8	97.48	**97.88**	-0.76
Orig	97.95	95.78	*98.16*	98.52	97.42	**-0.74**
Q-Orig	97.24	95.85	97.00	*98.15*	96.70	-1.45

Table 14: F1 Score of requested slot classification subtask with different description styles on unseen services. We train the model on SG-DST dataset for the description style in each row, then evaluate on 4 different descriptions styles. The mean are average performance of the remaining 3 descriptions styles. The Δ means the performance gap between the mean and the homogeneous performance

When creating MultiWOZ 2.2 (Zang et al., 2020), the slots with less than 50 different slot values are classified as categorical slots. We noticed that this leads inconsistent results with SG-DST dataset. It is hard to draw a consistent conclusion on the two datasets. According to the definition, we believe SG-DST are more suitable for categorical slot subtasks, we can further verify our guess when more datasets are created for the research of schema-guided dialog in the future.

Non-categorical Slot. We conduct non-categorical slot identification sub-tasks on both SG-DST and MultiWOZ 2.2 dataset. The results are shown in Table 16. Overall, the rich description performs better on both homogeneous and heterogeneous evaluations.

A.5.4 Qualitative Analysis On Heterogeneous Evaluation

We conduct qualitative analysis on heterogeneous evaluation on named-based description. Table 17 shows how paraphrasing the named-based description impact on the categorical and non-categorical slot prediction tasks.

The first 3 rows at the top are showing the cases of adding modifiers to the name. When the added extra modifiers are keywords in other slots, e.g. "attraction" are the keywords also used in "attraction_name". The first shows "attraction_location" may wrongly predicted as "attraction_name". It seems the model does not understand the compound nouns well, and they seems just pay attention to each key words "attraction" and "movie" here.

The 3 rows in the middle are showing the cases of using synonyms. Changing "to" to "target", and changing "movie" to "film" will cause extra confusion, which shows the model may fail to the synonyms.

The last 4 rows at the bottom is showing using abbreviations. Changing "number" to "num" will not impact the model, while changing "subtitle" to "sub" may let the model miss the key meaning of subtitle. The performance drop in the later case may be due to the misuse of the "sub" prefix, in En-

Style	*NameOnly*	*Q-Name*	*Orig*	*Q-Orig*	mean	Δ
			SG-DST			
NameOnly	*68.09*	58.41	63.49	62.21	61.37	-6.72
Q-Name	69.01	*68.29*	68.53	68.12	68.55	**+0.26**
Orig	70.19	65.91	*68.88*	69.64	**68.58**	-0.30
Q-Orig	69.98	65.97	69.26	*71.29*	68.40	-2.89
			MultiWOZ 2.2			
NameOnly	*59.24*	59.32	59.12	59.29	59.24	0.00
Q-Name	58.64	*59.74*	58.49	59.43	58.85	-0.89
Orig	59.26	59.91	*56.49*	58.97	**59.38**	**+2.89**
Q-Orig	60.00	60.70	51.18	*58.95*	57.29	-1.66

Table 15: Joint accuracy of categorical slot Subtask with different description styles on unseen services. Train the model on SG-DST and MultiWOZ 2.2 datasets respectively for each description style in each row, then evaluate on all 4 descriptions styles. The mean are the average performance of the remaining 3 descriptions styles. The Δ means the performance gap between the mean and the homogeneous performance

Style	*NameOnly*	*Q-Name*	*Orig*	*Q-Orig*	mean	Δ
			SG-DST			
NameOnly	*71.21*	49.85	59.8	59.95	56.53	-14.68
Q-Name	66.32	*69.22*	61.67	60.77	62.92	-6.30
Orig	78.73	51.57	*69.84*	69.87	**66.72**	**-3.12**
Q-Orig	62.6	36.44	69.49	*71.18*	56.18	-15.00
			MultiWOZ 2.2			
NameOnly	*61.30*	57.88	61.51	64.05	61.15	-0.15
Q-Name	60.62	*60.47*	60.6	62.58	61.27	+0.80
Orig	61.77	65.4	*61.39*	62.4	**63.19**	**+1.80**
Q-Orig	61.29	60.6	62.46	*62.45*	61.45	-1.00

Table 16: Joint accuracy of non-categorical slot Subtask with different description styles on unseen services. We train the model on SG-DST and MultiWOZ 2.2 datasets respectively for the description style in each row, then evaluate on all 4 different descriptions styles. The mean are the average performance of the remaining 3 descriptions styles. The Δ means the performance gap between the mean and the homogeneous performance

Service Name	Original Name	Paraphrased Name	Extra impact by the paraphrased name
Travel_1	location	attraction_location	Confused with other "attraction" prefixed slots, e.g. attraction _name
Movies_1	genre	movie_genre	Confused with movie_name
Movies_1	price	ticket_price, total_price	No impact
Bueses_3	to_city	target_city	The synonyms "target" is not understood well by model, confused with from_city
Movies_1	movie_name	film_name	The synonyms "film" is not understood well, getting wrong with theather_name
Hotels_2	where_to	house_loc	Improved by specific "house" keywords
Flights_4	origin_airport	orig_city_airport	More frequently predicted to slot "destination_airport"
Flights_4	destination_airport	dest_city_airport	More frequently predicted to slot "origin_airport"
Media_3	subtitle_language	sub_lang	Missing keyword "subtitle" make the slot inactive
Flights_4	number_of_tickets	num_of_tickets	No impact

Table 17: We analyze the confusion matrix of above slots before and after using the paraphrased name. We summarize the extra impact for using each paraphrased name.

glish, it usually means "secondary, less important, parts". We also found the "orig" and "dest" abbreviations may also understand well by the model. The above abbreviations seems reasonable paraphrases people will use for naming, while the are not understood well in the given context. Hence, in the design of schema-guided dialog, if using named-based description, we should be careful for about abbreviations used in the naming.