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 benchmarking descriptions and reveal the model robustness on both homogeneous and heterogeneous description styles in training and evaluation.
results, current schema-guided dialog state tracking 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 noncategorical 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 noncategorical slot labeling (Non-Cat). 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, Noncategorical, 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.

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 noncategorical 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 understand 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 their formats into a comparative discussion on different description styles §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 curpus 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 testbed for schema-guided dialog, which contains well-designed heterogenous 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 heterogenous evaluation 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

²Please refer to the original paper for more details.
Datasets  | Splits   | Dialog Domains | Services | Zero-shot Domains | Zero-shot Services | Function Overlap | Collecting Method |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SG-DST</td>
<td>Train</td>
<td>16 142</td>
<td>16 26</td>
<td>-</td>
<td>-</td>
<td>Across-domain</td>
<td>M2M</td>
</tr>
<tr>
<td></td>
<td>Dev</td>
<td>2482</td>
<td>16 17</td>
<td>1</td>
<td>8</td>
<td>Within-domain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>4201</td>
<td>18 21</td>
<td>3</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MULTIWOZ 2.2</td>
<td>Train</td>
<td>961 7</td>
<td>3 3</td>
<td>-</td>
<td>-</td>
<td>Across-domain</td>
<td>H2H</td>
</tr>
<tr>
<td></td>
<td>Dev</td>
<td>2455</td>
<td>5 5</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>2969</td>
<td>8 8</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

hotel room or travelling between cities. While SG-DST covers more domains, such as ‘Payments’, ‘Calender’, ‘DoctorServices’ and so on.

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 summerize the above discussion in Table 1. We believe that generalized results derived from these two 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

\(^3\) We use BERT-base-cased for all main experiments. Other pretrained language models can be easily adapted to our study

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 \(CLS_S\). Then for sentence-level representation, we concatenate dialog history representation \(CLS_D\) and candidate schema representation \(CLS_S\) as the whole sentence-level representation for the pair, denoted as \(CLS^{DE}\). For token-level representation, we concatenate the candidate schema \(CLS_S\) with each token embedding in the dialog history, denoted as \(TOK^{DE}\). Because the candidate schema em-
beddings are encoded independently from the dialog 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 CL$S^{CE}$ and each schema-attended dialog token embeddings 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 Tok$^{FE}$ including a schema-attended CL$S^{FE}$ 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 CL$S$ 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 CL$S$ representation to a single number $P^a_{I_k}$ with a linear layer follows a sigmoid function. We predict "NONE" if the $P^a_{I_k}$ of all intents are less than a threshold 0.5, which means no intent is active. Otherwise, we predict the intent with largest $P^a_{I_k}$. We predict the intent for each turn independently without considering the prediction on previous turns.

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

**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 CL$S$ 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;

**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 $f^s_{i}$ and $f^e_{i}$ 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 Mul-
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 study 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. In the rest of the paper, to focus more on intent/slot description, we do not use service description if not mentioned.

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.

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.

6 Supplementary Training (Q2)

Besides the pretrain-finetune 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. 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.

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 vs 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.

6.1 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.

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.

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.

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.
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 mostly 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 SQuAD 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 classifiers, which is different from the the single CLS classifier 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 homogenous 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:

- Identifier. 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 lowerbound 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 in-
query 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 make sure it retain the main meaning. Appendix A.5.1 shows examples for different styles of schema descriptions.

### 7.2 Results on Description Styles

For each style, we train separate models on 4 sub-tasks, 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 as shown in Table 7.

<table>
<thead>
<tr>
<th>Style/Task</th>
<th>SG-DST</th>
<th></th>
<th></th>
<th></th>
<th>MULTIWOZ 2.2</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Intent</td>
<td>Req</td>
<td>Cat</td>
<td>NonCat</td>
<td>Cat</td>
<td>NonCat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>94.24</td>
<td>98.86</td>
<td>74.01</td>
<td>74.86</td>
<td>64.19</td>
<td>56.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NameOnly</td>
<td>93.31</td>
<td>98.55</td>
<td>74.36</td>
<td>74.86</td>
<td>53.72</td>
<td>56.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q-Name</td>
<td>92.01</td>
<td>98.55</td>
<td>74.36</td>
<td>74.86</td>
<td>53.61</td>
<td>57.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Orig</td>
<td>94.24</td>
<td>98.86</td>
<td>74.01</td>
<td>74.86</td>
<td>64.19</td>
<td>56.17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q-Orig</td>
<td>93.31</td>
<td>98.55</td>
<td>74.36</td>
<td>74.86</td>
<td>53.72</td>
<td>56.18</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Homogeneous evaluation results of different 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, Identifier 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 meaning-

ful 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. 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 transactional 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. Additionally, Req is a relatively simpler task, requesting information are related to specific attributes, such as "has_live_music", "has_wifi", where keywords co-occurred 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, slots 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 that 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 similiar 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.

---

8Only 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.

9This action-object template has also been found efficient for open domain intent induction task (e.g., Vedula et al., 2020, OPINE).
Table 6: Performance changes when using BERT fine-tuned on SQuAD2 dataset to further finetuning on our NonCat task.

<table>
<thead>
<tr>
<th>Style/Dataset</th>
<th>all</th>
<th>seen</th>
<th>unseen</th>
<th>all</th>
<th>seen</th>
<th>unseen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig</td>
<td>+1.99</td>
<td>-1.79</td>
<td>+3.25</td>
<td>+1.93</td>
<td>-2.21</td>
<td>+4.27</td>
</tr>
<tr>
<td>Q-Orig</td>
<td>+6.13</td>
<td>-2.01</td>
<td>+8.84</td>
<td>+1.06</td>
<td>-1.28</td>
<td>+3.06</td>
</tr>
<tr>
<td>NameOnly</td>
<td>-0.45</td>
<td>-1.49</td>
<td>-0.11</td>
<td>+1.75</td>
<td>+0.58</td>
<td>+1.77</td>
</tr>
<tr>
<td>Q-Name</td>
<td>+0.05</td>
<td>-2.98</td>
<td>+1.04</td>
<td>-0.04</td>
<td>-0.32</td>
<td>+1.25</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Style/Task</th>
<th>Intent(Acc)</th>
<th>Req(F1)</th>
<th>Cat/Joint Acc</th>
<th>NonCat/Joint Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>mean</td>
<td>mean</td>
<td>mean</td>
<td>mean</td>
</tr>
<tr>
<td>NameOnly</td>
<td>82.93</td>
<td>11.47</td>
<td>96.92</td>
<td>-3.64</td>
</tr>
<tr>
<td>Orig</td>
<td>79.47</td>
<td>12.70</td>
<td>97.42</td>
<td>-0.74</td>
</tr>
<tr>
<td>Q-Orig</td>
<td>84.57</td>
<td>8.24</td>
<td>96.70</td>
<td>-1.45</td>
</tr>
<tr>
<td>para</td>
<td>para</td>
<td>para</td>
<td>para</td>
<td>para</td>
</tr>
<tr>
<td>NameOnly</td>
<td>92.32</td>
<td>-1.74</td>
<td>97.69</td>
<td>-0.87</td>
</tr>
<tr>
<td>Orig</td>
<td>91.54</td>
<td>-0.63</td>
<td>98.42</td>
<td>+0.26</td>
</tr>
</tbody>
</table>

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 a 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 benefit 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 covered 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 homogenous and heterogenous 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


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 Statistics 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.

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

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 lowerbound 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 summerize the functions 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.

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 noncategorical 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),

<table>
<thead>
<tr>
<th>Model/Task</th>
<th>SG-DST</th>
<th>MultiWOZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req Cat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonCat Cat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Seen Service</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>92.76</td>
<td>99.70</td>
</tr>
<tr>
<td>I/S Desc</td>
<td>95.35</td>
<td>99.74</td>
</tr>
<tr>
<td>Service + I/S Desc</td>
<td>95.28</td>
<td>99.74</td>
</tr>
<tr>
<td>Unseen Service</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>50.03</td>
<td>85.77</td>
</tr>
<tr>
<td>I/S Desc</td>
<td>92.17</td>
<td>98.16</td>
</tr>
<tr>
<td>Service + I/S Desc</td>
<td>86.95</td>
<td>97.99</td>
</tr>
</tbody>
</table>

Table 9: Models using different composition of schema, results on test set of SG-DST and our remixed MultiWOZ 2.2.
Table 10: Results of different supplementary training on SG-DST and MultiWOZ 2.2 dataset

<table>
<thead>
<tr>
<th>Style</th>
<th>Intent Description</th>
<th>Slot Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameOnly</td>
<td>CheckBalance</td>
<td>account_type</td>
</tr>
<tr>
<td>Q-Name</td>
<td>Is the user intending to CheckBalance?</td>
<td></td>
</tr>
<tr>
<td>Q-Orig</td>
<td>Check the amount of money in a user’s bank account</td>
<td>The account type of the user</td>
</tr>
<tr>
<td>Name-Para</td>
<td>CheckAccountBalance</td>
<td>user_account_type</td>
</tr>
<tr>
<td>Orig-Para</td>
<td>Check the balance of the user’s bank account</td>
<td>Type of the user account</td>
</tr>
</tbody>
</table>

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 Heterogeneous 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 applying SQuAD2-finetuned BERT on our models.

A.5.3 More Results On Homogenous and Heterogenous Evaluation

We list the detailed results for our evaluation across different styles. We use italic to show the homogenous evaluation, where the results are shown in the diagonal of each table, and we underline the best homogenous results in the diagonal. We use bold to show the best heterogenous 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
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 $\Delta$ means the performance gap between the mean and the homogeneous performance.

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 $\Delta$ means the performance gap between the mean and the homogeneous performance.

**Categorical Slot.** The results on SG-DST and MULTIWOZ 2.2 dataset are shown in Table 15. 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 15. Overall, the rich description performs better on both homogenous and heteroguous evolutions.

Table 15: Joint accuracy of categorical slot Sub-task 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 $\Delta$ means the performance gap between the mean and the homogeneous performance.

Table 16: Joint accuracy of non-categorical slot Sub-task 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 $\Delta$ means the performance gap between the mean and the homogeneous performance.