The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)

Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, Pranav Khaitan
Google Research, Mountain View, California, USA
{abhirast, xiaoxuez, srinivasksun, raghavgupta, pranavkhaitan}@google.com

Abstract

Virtual assistants such as Google Assistant, Alexa and Siri provide a conversational interface to a large number of services and APIs spanning multiple domains. Such systems need to support an ever-increasing number of services with possibly overlapping functionality. Furthermore, some of these services have little to no training data available. Existing public datasets for task-oriented dialogue do not sufficiently capture these challenges since they cover few domains and assume a single static ontology per domain. In this work, we introduce the the Schema-Guided Dialogue (SGD) dataset, containing over 16k multi-domain conversations spanning 16 domains. Our dataset exceeds the existing task-oriented dialogue corpora in scale, while also highlighting the challenges associated with building large-scale virtual assistants. It provides a challenging testbed for a number of tasks including language understanding, slot filling, dialogue state tracking and response generation. Along the same lines, we present a schema-guided paradigm for task-oriented dialogue, in which predictions are made over a dynamic set of intents and slots, provided as input, using their natural language descriptions. This allows a single dialogue system to easily support a large number of services and facilitates simple integration of new services without requiring additional training data. Building upon the proposed paradigm, we release a model for dialogue state tracking capable of zero-shot generalization to new APIs, while remaining competitive in the regular setting.

1 Introduction

Virtual assistants help users accomplish tasks including but not limited to finding flights, booking restaurants and, more recently, navigating user interfaces, by providing a natural language interface to services and APIs on the web. The recent popularity of conversational interfaces and the advent of frameworks like Actions on Google and Alexa Skills, which allow developers to easily add support for new services, has resulted in a major increase in the number of application domains and individual services that assistants need to support, following the pattern of smartphone applications.

Consequently, recent work has focused on scalable dialogue systems that can handle tasks across multiple application domains. Data-driven deep learning based approaches for multi-domain modeling have shown promise, both for end-to-end and modular systems involving dialogue state tracking and policy learning. This line of work has been facilitated by the release of multi-domain dialogue corpora such as MultiWOZ (Budzianowski et al. 2018), M2M (Shah et al. 2018) and FRAMES (El Asri et al. 2017).

However, existing datasets for multi-domain task-oriented dialogue do not sufficiently capture a number of challenges that arise with scaling virtual assistants in production. These assistants need to support a large (Kim et al. 2018), constantly increasing number of services over a large number of domains. In comparison, existing public datasets cover few domains. Furthermore, they define a single static API per domain, whereas multiple services with overlapping functionality, but heterogeneous interfaces, exist in the real world.

To highlight these challenges, we introduce the Schema-Guided Dialogue (SGD) dataset[1], which is, to the best of our knowledge, the largest public task-oriented dialogue corpus. It exceeds existing corpora in scale, with over 16000 dialogues in the training set spanning 26 services belonging to 16 domains (more details in Table 1). Further, to adequately test the models' ability to generalize in zero-shot settings, the evaluation sets contain unseen services and domains. The dataset is designed to serve as an effective testbed for intent prediction, slot filling, state tracking and language generation, among other tasks in large-scale virtual assistants.

We also propose the schema-guided paradigm for task-oriented dialogue, advocating building a single unified dialogue model for all services and APIs. Using a service's schema as input, the model would make predictions over this dynamic set of intents and slots present in the schema. This setting enables effective sharing of knowledge among all services, by relating semantically similar concepts across APIs, and allows the model to handle unseen services and APIs. Under the proposed paradigm, we present a novel architecture for multi-domain dialogue state tracking. By using large pre-trained models like BERT (Devlin et al. 2019), our model can generalize to unseen services and is robust to API changes, while achieving competitive results on the original and updated MultiWOZ datasets (Eric et al. 2019).

[1]The dataset has been released at github.com/google-research-datasets/dstc8-schema-guided-dialogue

Metric ↓ Dataset →	DSTC2	WOZ2.0	FRAMES	M2M	MultiWOZ	SGD
No. of domains	1	1	3	2	7	**16**
No. of dialogues	1,612	600	1,369	1,500	8,438	**16,142**
Total no. of turns	23,354	4,472	19,986	14,796	113,556	**329,964**
Avg. turns per dialogue	14.49	7.45	14.60	9.86	13.46	**20.44**
Avg. tokens per turn	8.54	11.24	12.60	8.24	**13.13**	9.75
Total unique tokens	986	2,142	12,043	1,008	23,689	**30,352**
No. of slots	8	4	61	13	24	**214**
No. of slot values	212	99	3,871	138	4,510	**14,139**

Table 1: Comparison of our SGD dataset to existing related datasets for task-oriented dialogue. Note that the numbers reported are for the training portions for all datasets except FRAMES, where the numbers for the complete dataset are reported.

2 Related Work

Task-oriented dialogue systems have constituted an active area of research for decades. The growth of this field has been consistently fueled by the development of new datasets. Initial datasets were limited to one domain, such as ATIS (Hemphill, Godfrey, and Doddington 1990) for spoken language understanding for flights. The Dialogue State Tracking Challenges (Williams et al. 2013; Henderson, Thomson, and Williams 2014a; 2014b; Kim et al. 2017) contributed to the creation of dialogue datasets with increasing complexity. Other notable related datasets include WOZ2.0 (Wen et al. 2017), FRAMES (El Asri et al. 2017), M2M (Shah et al. 2018) and MultiWOZ (Budzianowski et al. 2018). These datasets have utilized a variety of data collection techniques, falling within two broad categories:

- **Wizard-of-Oz** This setup (Kelley 1984) connects two crowd workers playing the roles of the user and the system. The user is provided a goal to satisfy, and the system accesses a database of entities, which it queries as per the user's preferences. WOZ2.0, FRAMES and MultiWOZ, among others, have utilized such methods.
- **Machine-machine Interaction** A related line of work explores simulation-based dialogue generation, where the user and system roles are simulated to generate a complete conversation flow, which can then be converted to natural language using crowd workers as done in Shah et al. (2018). Such a framework may be cost-effective and error-resistant since the underlying crowd worker task is simpler, and annotations are obtained automatically.

As virtual assistants incorporate diverse domains, recent work has focused on zero-shot modeling (Bapna et al. 2017; Xia et al. 2018; Shah et al. 2019), domain adaptation and transfer learning techniques (Yang, Salakhutdinov, and Cohen 2017; Rastogi, Hakkani-Tür, and Heck 2017; Zhu and Yu 2018). Deep-learning based approaches have achieved state of the art performance on dialogue state tracking tasks. Popular approaches on small-scale datasets estimate the dialogue state as a distribution over all possible slot-values (Henderson, Thomson, and Young 2014; Wen et al. 2017; Ren et al. 2018) or individually score all slot-value combinations (Mrkšić et al. 2017; Zhong, Xiong, and Socher 2018). Such approaches are not practical for deployment in virtual assistants operating over real-world services having a very large and dynamic set of possible values. Addressing these concerns, approaches utilizing a dynamic vocabulary of slot values have been proposed (Rastogi, Gupta, and Hakkani-Tur 2018; Goel, Paul, and Hakkani-Tür 2019; Wu et al. 2019).

3 The Schema-Guided Dialogue Dataset

An important goal of this work is to create a benchmark dataset highlighting the challenges associated with building large-scale virtual assistants. Table 1 compares our dataset with other public datasets. Our Schema-Guided Dialogue (SGD) dataset exceeds other datasets in most of the metrics at scale. The especially larger number of domains, slots, and slot values, and the presence of multiple services per domain, are representative of these scale-related challenges. Furthermore, our evaluation sets contain many services, and consequently slots, which are not present in the training set, to help evaluate model performance on unseen services.

The 20 domains present across the train, dev and test splits are listed in Table 2. We create synthetic implementations of a total of 45 services or APIs over these domains. Our simulator framework interacts with these services to generate dialogue outlines, which are a structured representation of dialogue semantics. We then used a crowd-sourcing procedure to paraphrase these outlines to natural language utterances. Our novel crowd-sourcing procedure preserves all annotations obtained from the simulator and does not require any extra annotations after dialogue collection. In this section, we describe these steps in detail and then present analyses of the collected dataset.

3.1 Services and APIs

We define the schema for a service as a combination of intents and slots with additional constraints, with an example in Figure 1. We implement all services using a SQL engine. For constructing the underlying tables, we sample a set of entities from Freebase and obtain the values for slots defined in the schema from the appropriate attribute in Freebase. We decided to use Freebase to sample real-world entities instead of synthetic ones since entity attributes are often correlated (e.g, a restaurant's name is indicative of the cuisine served). Some slots like event dates/times and available ticket counts, which are not present in Freebase, are synthetically sampled.

To reflect the constraints present in real-world services and APIs, we impose a few other restrictions. First, our dataset does not expose the set of all possible slot values

Domain	#Intents	#Dialogs	Domain	#Intents	#Dialogs
Alarm[2,3]	2 (1)	324	Movies[1,2,3]	5 (3)	2339
Banks[1,2]	4 (2)	1021	Music[1,2,3]	6 (3)	1833
Buses[1,2,3]	6 (3)	3135	Payment[3]	2 (1)	222
Calendar[1]	3 (1)	1602	RentalCars[1,2,3]	6 (3)	2510
Events[1,2,3]	7 (3)	4519	Restaurants[1,2,3]	4 (2)	3218
Flights[1,2,3]	10 (4)	3644	RideSharing[1,2,3]	2 (2)	2223
Homes[1,2,3]	2 (1)	1273	Services[1,2,3]	8 (4)	2956
Hotels[1,2,3]	8 (4)	4992	Train[3]	2 (1)	350
Media[1,2,3]	6 (3)	1656	Travel[1,2,3]	1 (1)	2808
Messaging[3]	1 (1)	298	Weather[1,2,3]	1 (1)	1783

Table 2: The total number of intents (services in parentheses) and dialogues for each domain across train[1], dev[2] and test[3] sets. Multi-domain dialogues contribute to counts of each constituent domain. The domain 'Service' includes salons, dentists, doctors etc. The 'Alarm', 'Messaging', 'Payment' and 'Train' domains are only present in the dev or test sets to test generalization to new domains.

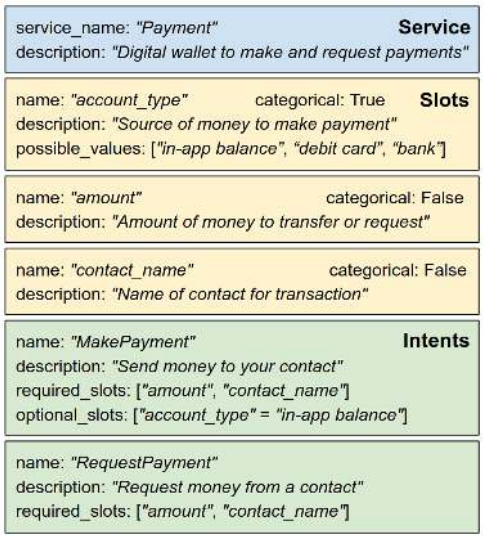


Figure 1: Example schema for a digital wallet service.

for some slots. Having such a list is impractical for slots like date or time because they have infinitely many possible values or for slots like movie or song names, for which new values are periodically added. Our dataset specifically identifies such slots as *non-categorical* and does not provide a set of all possible values for these. We also ensure that the evaluation sets have a considerable fraction of slot values not present in the training set to evaluate the models in the presence of new values. Some slots like gender, number of people, day of the week etc. are defined as *categorical* and we specify the set of all possible values taken by them. However, these values are not assumed to be consistent across services. E.g., different services may use ('male', 'female'), ('M', 'F') or ('he', 'she') as possible values for gender slot.

Second, real-world services can only be invoked with a limited number of slot combinations: e.g. restaurant reservation APIs do not let the user search for restaurants by date without specifying a location. However, existing datasets simplistically allow service calls with any given combination of slot values, thus giving rise to flows unsupported by actual services or APIs. As in Figure 1, the different service calls supported by a service are listed as intents. Each intent specifies a set of required slots and the system is not allowed to call this intent without specifying values for these required slots. Each intent also lists a set of optional slots with default values, which the user can override.

3.2 Dialogue Simulator Framework

The dialogue simulator interacts with the services to generate dialogue outlines. Figure 2 shows the overall architecture of our dialogue simulator framework. It consists of two agents playing the roles of the user and the system. Both agents interact with each other using a finite set of actions specified through dialogue acts over a probabilistic automaton designed to capture varied dialogue trajectories. These dialogue acts can take a slot or a slot-value pair as argument. Figure 4b shows all dialogue acts supported by the agents.

At the start of a conversation, the user agent is seeded with a scenario, which is a sequence of intents to be fulfilled. We identified over 200 distinct scenarios for the training set, each comprising up to 5 intents. For multi-domain dialogues, we also identify combinations of slots whose values may be transferred when switching intents e.g. the *'address'* slot value in a restaurant service could be transferred to the *'destination'* slot for a taxi service invoked right after.

The user agent then generates the dialogue acts to be output in the next turn. It may retrieve arguments i.e. slot values for some of the generated acts by accessing either the service schema or the SQL backend. The acts, combined with the respective parameters yield the corresponding user actions. Next, the system agent generates the next set of actions using a similar procedure. Unlike the user agent, however, the system agent has restricted access to the services (denoted by dashed line), e.g. it can only query the services by supplying values for all required slots for some service call. This helps us ensure that all generated flows are valid.

After an intent is fulfilled through a series of user and system actions, the user agent queries the scenario to proceed to the next intent. Alternatively, the system may suggest related intents e.g. reserving a table after searching for a restaurant. The simulator also allows for multiple intents to be active during a given turn. While we skip many implementation details for brevity, it is worth noting that we do not include any domain-specific constraints in the simulation automaton. All domain-specific constraints are encoded in the schema and scenario, allowing us to conveniently use the simulator across a wide variety of domains and services.

3.3 Dialogue Paraphrasing

The dialogue paraphrasing framework converts the outlines generated by the simulator into a natural conversation. Figure 3a shows a snippet of the dialogue outline generated by the simulator, containing a sequence of user and system actions. The slot values present in these actions are in a canonical form because they obtained directly from the service. However, users may refer to these values in various differ-

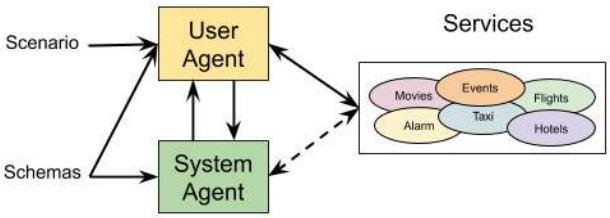


Figure 2: The overall architecture of the dialogue simulation framework for generating dialogue outlines.

ent ways during the conversation, e.g., "los angeles" may be referred to as "LA" or "LAX". To introduce these natural variations in the slot values, we replace different slot values with a randomly selected variation (kept consistent across user turns in a dialogue) as shown in Figure 3b.

Next we define a set of action templates for converting each action into a utterance. A few examples of such templates are shown below. These templates are used to convert each action into a natural language utterance, and the resulting utterances for the different actions in a turn are concatenated together as shown in Figure 3c. The dialogue transformed by these steps is then sent to the crowd workers. One crowd worker is tasked with paraphrasing all utterances of a dialogue to ensure naturalness and coherence.

`REQUEST(location)` → Which city are you in?
`INFORM(location=$x)` → I want to eat in $x.
`OFFER(restaurant=$x)` → $x is a nice restaurant.

In our paraphrasing task, the crowd workers are instructed to exactly repeat the slot values in their paraphrases. This not only helps us verify the correctness of the paraphrases, but also lets us automatically obtain slot spans in the generated utterances by string search. This automatic slot span generation greatly reduced the annotation effort required, with little impact on dialogue naturalness, thus allowing us to collect more data with the same resources. Furthermore, it is important to note that this entire procedure preserves all other annotations obtained from the simulator including the dialogue state. Hence, no further annotation is needed.

3.4 Dataset Analysis

With over 16000 dialogues in the training set, the Schema-Guided Dialogue dataset is the largest publicly available annotated task-oriented dialogue dataset. The annotations include the active intents and dialogue states for each user utterance and the system actions for every system utterance. We have a few other annotations like the user actions but we withhold them from the public release. These annotations enable our dataset to be used as benchmark for tasks like intent detection, dialogue state tracking, imitation learning of dialogue policy, dialogue act to text generation etc. The schemas contain semantic information about the APIs and the constituent intents and slots, in the form of natural language descriptions and other details (example in Figure 1).

The single-domain dialogues in our dataset contain an average of 15.3 turns, whereas the multi-domain ones contain

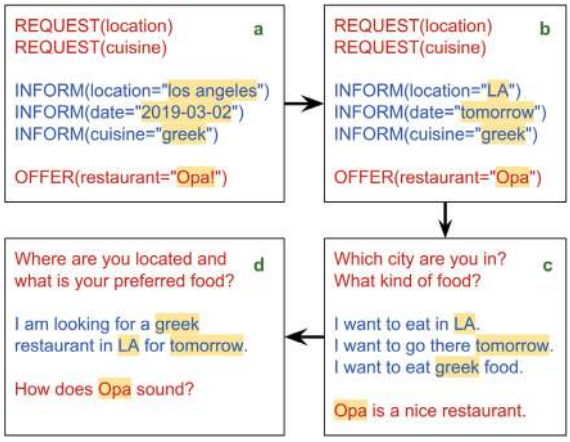


Figure 3: Steps for obtaining paraphrased conversations. To increase the presence of relative dates like tomorrow, next Monday, the current date is assumed to be March 1, 2019.

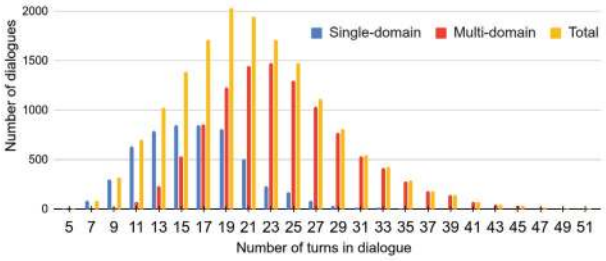
23 turns on an average. These numbers are also reflected in Figure 4a showing the histogram of dialogue lengths on the training set. Table 2 shows the distribution of dialogues across the different domains. We note that distribution of dialogues across the domains and services covered is largely balanced, with the exception domains which are not present in the training set. Figure 4b shows the frequency of dialogue acts contained in the dataset. Note that all dialogue acts except `INFORM`, `REQUEST` and `GOODBYE` are specific to either the user or the system.

4 The Schema-Guided Approach

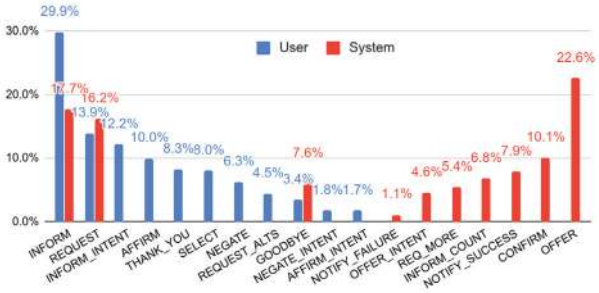
Virtual assistants aim to support a large number of services available on the web. One possible approach is to define a large unified schema for the assistant, to which different service providers can integrate with. However, it is difficult to come up with a common schema covering all use cases. Having a common schema also complicates integration of tail services with limited developer support. We propose the schema-guided approach as an alternative to allow easy integration of new services and APIs.

Under our proposed approach, each service provides a schema listing the supported slots and intents along with their natural language descriptions (Figure 1 shows an example). These descriptions are used to obtain a semantic representation of these schema elements. The assistant employs a single unified model containing no domain or service specific parameters to make predictions conditioned on these schema elements. For example, Figure 5 shows how dialogue state representation for the same dialogue can vary for two different services. Here, the departure and arrival cities are captured by analogously functioning but differently named slots in both schemas. Furthermore, values for the *number_stops* and *direct_only* slots highlight idiosyncrasies between services interpreting the same concept.

Using a single model facilitates representation and transfer of common knowledge across related services. Since the model utilizes semantic representation of schema elements



(a) Histogram of lengths of training set dialogues.



(b) Distribution of dialogue acts in training set.

Figure 4: Detailed statistics of the SGD dataset.

as input, it can interface with unseen services or APIs on which it has not been trained. It is also robust to changes like addition of new intents or slots to the service.

5 Zero-Shot Dialogue State Tracking

Models in the schema-guided setting can condition on the pertinent services' schemas using descriptions of intents and slots. These models, however, also need access to representations for potentially unseen inputs from new services. Recent pretrained models like ELMo (Peters et al. 2018) and BERT (Devlin et al. 2019) can help, since they are trained on very large corpora. Building upon these, we present a simple prototype model for zero-shot schema-guided DST.

5.1 Model

We use a single model[2], shared among all services and domains, to make these predictions. We first encode all the intents, slots and slot values for categorical slots present in the schema into an embedded representation. Since different schemas can have differing numbers of intents or slots, predictions are made over dynamic sets of schema elements by conditioning them on the corresponding schema embeddings. This is in contrast to existing models which make predictions over a static schema and are hence unable to share knowledge across domains and services. They are also not robust to changes in schema and require the model to be retrained with new annotated data upon addition of a new intent, slot, or in some cases, a slot value to a service.

Schema Embedding This component obtains the embedded representations of intents, slots and categorical slot values in each service schema. Table 3 shows the sequence pairs used for embedding each schema element. These sequence pairs are fed to a pretrained BERT encoder shown in Figure 6 and the output $\mathbf{u}_{\texttt{CLS}}$ is used as the schema embedding.

	Sequence 1	Sequence 2
Intent	service description	intent description
Slot	service description	slot description
Value	slot description	value

Table 3: Input sequences for the pretrained BERT model to obtain embeddings of different schema elements.

For a given service with I intents and S slots, let $\{\mathbf{i}_j\}$, $1 \leq j \leq I$ and $\{\mathbf{s}_j\}$, $1 \leq j \leq S$ be the embeddings of all intents and slots respectively. As a special case, we let $\{\mathbf{s}^n_j\}$, $1 \leq j \leq N \leq S$ denote the embeddings for the N non-categorical slots in the service. Also, let $\{\mathbf{v}^k_j\}$, $1 \leq j \leq V^k$ denote the embeddings for all possible values taken by the k^{th} categorical slot, $1 \leq k \leq C$, with C being the number of categorical slots and $N + C = S$. All these embeddings are collectively called schema embeddings.

Utterance Encoding Like Chao and Lane (2019), we use BERT to encode the user utterance and the preceding system utterance to obtain utterance pair embedding $\mathbf{u} = \mathbf{u}_{\texttt{CLS}}$ and token level representations $\mathbf{t}_1, \mathbf{t}_2 \cdots \mathbf{t}_M$, M being the total number of tokens in the two utterances. The utterance and schema embeddings are used together to obtain model predictions using a set of projections (defined below).

Projection Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. For a task K, we define $\mathbf{l} = \mathcal{F}_K(\mathbf{x}, \mathbf{y}, p)$ as a projection transforming $\mathbf{x}$ and $\mathbf{y}$ into the vector $\mathbf{l} \in \mathbb{R}^p$ using Equations 1-3. Here, $\mathbf{h_1}, \mathbf{h_2} \in \mathbb{R}^d$, W^K_i and b^K_i for $1 \leq i \leq 3$ are trainable parameters of suitable dimensions and A is the activation function. We use `gelu` (Hendrycks and Gimpel 2016) activation as in BERT.

$$\mathbf{h_1} = A(W^K_1 \mathbf{x} + b^K_1) \quad (1)$$

$$\mathbf{h_2} = A(W^K_2 (\mathbf{y} \oplus \mathbf{h_1}) + b^K_2) \quad (2)$$

$$\mathbf{l} = W^K_3 \mathbf{h_2} + b^K_3 \quad (3)$$

Active Intent For a given service, the active intent denotes the intent requested by the user and currently being fulfilled by the system. It takes the value "NONE" if no intent for the service is currently being processed. Let $\mathbf{i}_0$ be a trainable parameter in $\mathbb{R}^d$ for the "NONE" intent. We define the intent network as below.

$$l^j_{\text{int}} = \mathcal{F}_{\text{int}}(\mathbf{u}, \mathbf{i}_j, 1), \;\; 0 \leq j \leq I \quad (4)$$

The logits l^j_{int} are normalized using softmax to yield a distribution over all I intents and the "NONE" intent. During inference, we predict the highest probability intent as active.

Requested Slots These are the slots whose values are requested by the user in the current utterance. Projection $\mathcal{F}_{\text{req}}$ predicts logit l^j_{req} for the j^{th} slot. Obtained logits are normalized using sigmoid to get a score in $[0, 1]$. During inference, all slots with score > 0.5 are predicted as requested.

$$l^j_{\text{req}} = \mathcal{F}_{\text{req}}(\mathbf{u}, \mathbf{s}_j, 1), \;\; 1 \leq j \leq S \quad (5)$$

[2]Our model code is available at github.com/google-research/google-research/tree/master/schema_guided_dst

Figure 5: The predicted dialogue state (shown with dashed edges) for the first two user turns for an example dialogue, showing the active intent and slot assignments, with two related annotation schemas. Note that the dialogue state representation is conditioned on the schema under consideration, which is provided as input, as are the user and system utterances.

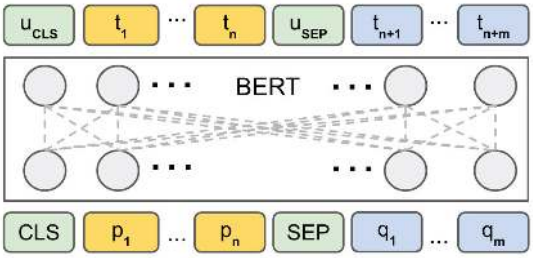


Figure 6: BERT encoder, taking in two sequences p and q as input and outputs an embedded sequence pair representation $\mathbf{u}_{\texttt{CLS}}$ and token level representations $\{\mathbf{t}_1 \cdots \mathbf{t}_{n+m}\}$. We use BERT to obtain schema element embeddings and encode system and user utterances for dialogue state tracking.

User Goal We define the user goal as the user constraints specified over the dialogue context till the current user utterance. Instead of predicting the entire user goal after each user utterance, we predict the difference between the user goal for the current turn and preceding user turn. During inference, the predicted user goal updates are accumulated to yield the predicted user goal. We predict the user goal updates in two stages. First, for each slot, a distribution of size 3 denoting the slot status and taking values `none`, `dontcare` and `active` is obtained by normalizing the logits obtained in equation 6 using softmax. If the status of a slot is predicted to be `none`, its assigned value is assumed to be unchanged. If the prediction is `dontcare`, then the special `dontcare` value is assigned to it. Otherwise, a slot value is predicted and assigned to it in the second stage.

$$\mathbf{l}^j_{\text{status}} = \mathcal{F}_{\text{status}}(\mathbf{u}, \mathbf{s}_j, 3), \;\; 1 \leq j \leq S \quad (6)$$

$$l^{j,k}_{\text{value}} = \mathcal{F}_{\text{value}}(\mathbf{u}, \mathbf{v}^k_j, 1), \;\; 1 \leq j \leq V^k, 1 \leq k \leq C \quad (7)$$

$$l^{j,k}_{\text{start}} = \mathcal{F}_{\text{start}}(\mathbf{t}_k, \mathbf{s}^n_j, 1), \;\; 1 \leq j \leq N, 1 \leq k \leq M \quad (8)$$

$$l^{j,k}_{\text{end}} = \mathcal{F}_{\text{end}}(\mathbf{t}_k, \mathbf{s}^n_j, 1), \;\; 1 \leq j \leq N, 1 \leq k \leq M \quad (9)$$

In the second stage, equation 7 is used to obtain a logit for each value taken by each categorical slot. Logits for a given categorical slot are normalized using softmax to get a distribution over all possible values. The value with the maximum mass is assigned to the slot. For each non-categorical slot, logits obtained using equations 8 and 9 are normalized using softmax to yield two distributions over all tokens. These two distributions respectively correspond to the start and end index of the span corresponding to the slot. The indices $p \leq q$ maximizing $start[p] + end[q]$ are predicted to be the span boundary and the corresponding value is assigned to the slot.

5.2 Evaluation

We consider the following metrics for evaluation of the dialogue state tracking task:

1. **Active Intent Accuracy:** The fraction of user turns for which the active intent has been correctly predicted.
2. **Requested Slot F1:** The macro-averaged F1 score for requested slots over all eligible turns. Turns with no requested slots in ground truth and predictions are skipped.
3. **Average Goal Accuracy:** For each turn, we predict a single value for each slot present in the dialogue state. The slots which have a non-empty assignment in the ground truth dialogue state are considered for accuracy. This is the average accuracy of predicting the value of a slot correctly. A fuzzy matching score is used for non-categorical slots to reward partial matches with the ground truth.
4. **Joint Goal Accuracy:** This is the average accuracy of predicting *all* slot assignments for a turn correctly. For non-categorical slots a fuzzy matching score is used.

Performance on other datasets We evaluate our model on public datasets WOZ2.0 and MultiWOZ 2.1 (Eric et al. 2019). As results in Table 4 show, our model performs competitively on these datasets. In these experiments, we omit the use of fuzzy matching scores and use exact match while calculating the goal accuracies to keep our numbers comparable with other works. Furthermore, for the MultiWOZ 2.1 dataset, we also trained a model incorporating pointer-generator style copying for non-categorical slots, similar to Wu et al. (2019), giving us a joint goal accuracy of **0.489**, exceeding the best-known result of 0.456 as reported in Eric et al. (2019). We omit the details of this model since it is not the main focus of this work.

Performance on SGD The model performs well for Active Intent Accuracy and Requested Slots F1 across both seen and unseen services, shown in Table 4. For joint goal and average goal accuracy, the model performs better on seen services compared to unseen ones (Figure 7). The main reason for this performance difference is a significantly higher OOV rate for slot values of unseen services.

Performance on different domains (SGD) The model performance also varies across various domains. The performance for the different domains is shown in Table 5. We observe that one of the major factors affecting the performance across domains is still the presence of the service in the training data (seen services). In most cases, the performance can be observed to degrade for domains with more unseen services. Among the unseen services, those in the 'RentalCars' and 'Buses' domain, have a very high OOV rate for slot values leading to worse performance. A low joint goal accuracy and high average goal accuracy for these two domains indicates a possible skew between the performance of different slots. Among seen services, 'RideSharing' domain also exhibits poor performance, since it possesses the largest number of the possible slot values across the dataset. We also notice that for categorical slots, with similar slot values (e.g. "Psychologist" and "Psychiatrist"), there is a very weak signal for the model to distinguish between the different classes, resulting in inferior performance.

Dataset	Active Int Acc	Req Slot F1	Avg GA	Joint GA
WOZ2.0	N.A.	0.970	0.920	0.810
MultiWOZ 2.1	N.A.	N.A.	0.875	0.434
SGD-S	0.885	0.956	0.684	0.356
SGD-All	0.906	0.965	0.560	0.254

Table 4: Model performance on test sets of the respective datasets. SGD-Single model is trained on single-domain dialogues only whereas SGD-All model is trained on the entire training set. We also report results on MultiWOZ 2.1 and WOZ2.0. N.A. indicates unavailable tasks.

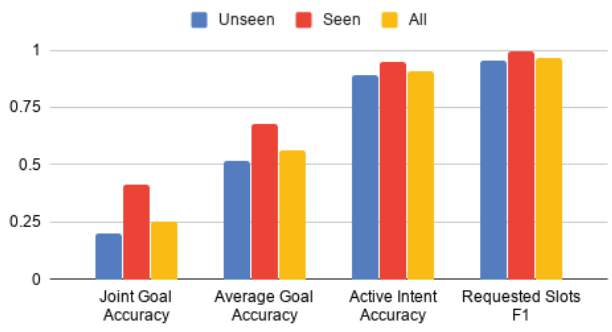


Figure 7: Performance of the model on all services, services seen in training data, services not seen in training data.

Domain	Joint GA	Avg GA	Domain	Joint GA	Avg GA
RentalCars*	0.086	0.480	Restaurants*	0.228	0.558
Buses*	0.097	0.509	Events*	0.235	0.579
Messaging*	0.102	0.200	Flights*	0.239	0.659
Payment*	0.115	0.348	Hotels**	0.289	0.582
Trains*	0.136	0.635	Movies**	0.378	0.686
Music*	0.155	0.399	Services**	0.409	0.721
RideSharing	0.170	0.502	Travel	0.415	0.572
Media*	0.180	0.308	Alarm*	0.577	0.018
Homes	0.189	0.727	Weather	0.620	0.764

Table 5: Model performance per domain (GA: goal accuracy). Domains marked with '*' are those for which the service in the test set is not present in the training set. Domains like Hotels marked with '**' has one unseen and one seen service. For other domains, the service in the test set was also seen in the training set. We see that the model generally performs better for domains containing services seen during training.

6 Discussion

It is often argued that simulation-based data collection does not yield natural dialogues or sufficient coverage, when compared to other approaches such as Wizard-of-Oz. We argue that simulation-based collection is a better alternative for collecting datasets like this owing to the factors below.

- **Fewer Annotation Errors:** All annotations are automatically generated, so these errors are rare. In contrast, Eric et al. (2019) reported annotation errors in 40% of turns in MultiWOZ 2.0 which utilized a Wizard-of-Oz setup.
- **Simpler Task:** The crowd worker task of paraphrasing a readable utterance for each turn is simple. The error-prone annotation task requiring skilled workers is not needed. Furthermore, Wizard-of-Oz style collection requires domain specific task definitions and instructions, making the collection of a diverse dataset like ours time consuming.
- **Low Cost:** The simplicity of the crowd worker task and lack of an annotation task greatly cut data collection costs.
- **Better Coverage:** A wide variety of dialogue flows can be collected and specific usecases can be targeted.

To ensure naturalness of the generated conversations, we used the conversational flows present in other public datasets like MultiWOZ 2.0 and WOZ2.0 as a guideline while developing the dialogue simulator. It was difficult for us to conduct a side-by-side comparison with existing datasets since this is the first dataset to cover many new domains at such scale, but we plan to explore it in the future.

7 Conclusions

We presented the Schema-Guided Dialogue dataset to encourage scalable modeling approaches for virtual assistants. We also introduced the schema-guided paradigm for task-oriented dialogue that simplifies the integration of new services and APIs with large scale virtual assistants. Building upon this paradigm, we present a simplistic model for zero-shot dialogue state tracking achieving competitive results.

Acknowledgements The authors thank Guan-Lin Chao, Amir Fayazi and Maria Wang for their advice and assistance.

References

Bapna, A.; Tür, G.; Hakkani-Tür, D.; and Heck, L. P. 2017. Towards zero-shot frame semantic parsing for domain scaling. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*.

Budzianowski, P.; Wen, T.-H.; Tseng, B.-H.; Casanueva, I.; Ultes, S.; Ramadan, O.; and Gasic, M. 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*, 5016–5026.

Chao, G.-L., and Lane, I. 2019. BERT-DST: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. In *INTERSPEECH*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 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)*, 4171–4186.

El Asri, L.; Schulz, H.; Sharma, S.; Zumer, J.; Harris, J.; Fine, E.; Mehrotra, R.; and Suleman, K. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, 207–219.

Eric, M.; Goel, R.; Paul, S.; Sethi, A.; Agarwal, S.; Gao, S.; and Hakkani-Tur, D. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Goel, R.; Paul, S.; and Hakkani-Tür, D. 2019. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *arXiv preprint arXiv:1907.00883*.

Hemphill, C. T.; Godfrey, J. J.; and Doddington, G. R. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

Henderson, M.; Thomson, B.; and Williams, J. D. 2014a. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 263–272.

Henderson, M.; Thomson, B.; and Williams, J. D. 2014b. The third dialog state tracking challenge. *2014 IEEE Spoken Language Technology Workshop (SLT)* 324–329.

Henderson, M.; Thomson, B.; and Young, S. 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)*, 292–299.

Hendrycks, D., and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

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

Kim, S.; D'Haro, L. F.; Banchs, R. E.; Williams, J. D.; and Henderson, M. 2017. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*. Springer. 435–449.

Kim, Y.-B.; Kim, D.; Kumar, A.; and Sarikaya, R. 2018. Efficient large-scale neural domain classification with personalized attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2214–2224. Melbourne, Australia: Association for Computational Linguistics.

Mrkšić, N.; Séaghdha, D. Ó.; Wen, T.-H.; Thomson, B.; and Young, S. 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)*, volume 1, 1777–1788.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Rastogi, A.; Gupta, R.; and Hakkani-Tur, D. 2018. Multi-task learning for joint language understanding and dialogue state tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, 376–384.

Rastogi, A.; Hakkani-Tür, D.; and Heck, L. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 561–568. IEEE.

Ren, L.; Xie, K.; Chen, L.; and Yu, K. 2018. Towards universal dialogue state tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2780–2786.

Shah, P.; Hakkani-Tür, D.; Tür, G.; Rastogi, A.; Bapna, A.; Nayak, N.; and Heck, L. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Shah, D.; Gupta, R.; Fayazi, A.; and Hakkani-Tur, D. 2019. Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 5484–5490. Florence, Italy: Association for Computational Linguistics.

Wen, T.; Vandyke, D.; Mrkšíc, N.; Gašíc, M.; Rojas-Barahona, L.; Su, P.; Ultes, S.; and Young, S. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017-Proceedings of Conference*, volume 1, 438–449.

Williams, J.; Raux, A.; Ramachandran, D.; and Black, A. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, 404–413.

Wu, C.-S.; Madotto, A.; Hosseini-Asl, E.; Xiong, C.; Socher, R.; and Fung, P. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 808–819. Florence, Italy: Association for Computational Linguistics.

Xia, C.; Zhang, C.; Yan, X.; Chang, Y.; and Yu, P. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 3090–3099. Association for Computational Linguistics.

Yang, Z.; Salakhutdinov, R.; and Cohen, W. W. 2017. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*.

Zhong, V.; Xiong, C.; and Socher, R. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1458–1467. Melbourne, Australia: Association for Computational Linguistics.

Zhu, S., and Yu, K. 2018. Concept transfer learning for adaptive language understanding. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, 391–399.