Hi-DST: A Hierarchical Approach for Scalable and Extensible Dialogue State Tracking

Suvodip Dey
Indian Institute of Technology,
Hyderabad, India
suvodip15@gmail.com

Maunendra Sankar Desarkar
Indian Institute of Technology,
Hyderabad, India
maunendra@cse.iith.ac.in

Abstract

Dialogue State Tracking (DST) is a sub-task of task-based dialogue systems where the user intention is tracked through a set of (domain, slot, slot-value) triplets. Existing DST models can be difficult to extend for new datasets with larger domains/slots mainly due to either of the two reasons- i) prediction of domain-slot as a pair, and ii) dependency of model parameters on the number of slots and domains. In this work, we propose to address these issues using a **Hi**erarchical **DST** (**Hi-DST**) model. At a given turn, the model first detects a change in domain followed by domain prediction if required. Then it decides suitable action for each slot in the predicted domains and finds their value accordingly. The model parameters of Hi-DST are independent of the number of domains/slots. Due to the hierarchical modeling, it achieves $O(|M| + |N|)$ belief state prediction for a single turn where M and N are the set of unique domains and slots respectively. We argue that the hierarchical structure helps in the model explainability and makes it easily extensible to new datasets. Experiments on the MultiWOZ dataset show that our proposed model achieves comparable joint accuracy performance to state-of-the-art DST models.

U_0: Can you help me find some attractions in the **east** part of town?
B_0: { (attraction, area, east) }

S_1 : Definitely! My favorite place in the east is the **Funky Fun House**. It's funky and fun!
U_1: Can I have the number please?
B_1: { (attraction, area, east), (attraction, name, Funky Fun House) }

S_2 : It's 01223304705. Do you need anything else?
U_2: Yeah, I need a restaurant. They need to serve **Indian** food and be in the **same area** as Funky Fun House.
B_2: { (attraction, area, east), (attraction, name, Funky Fun House), (restaurant, area, east), (restaurant, food, Indian) }

S_3 : There are 4 Indian restaurants in the area. Two are moderately priced and two are expensive. Can I ask what price range you would like?
U_3: I would prefer one in the **moderate** price range.
B_3: { (attraction, area, east), (attraction, name, Funky Fun House), (restaurant, area, east), (restaurant, food, Indian), (restaurant, price, moderate) }

S_4 : May I suggest the **Rajmahal** located at 7 Barnwell Road Fen Ditton.
U_4: Can I also have their phone number and postcode?
B_4 : { (attraction, area, east), (attraction, name, Funky Fun House), (restaurant, area, east), (restaurant, food, Indian), (restaurant, price, moderate), (restaurant, name, Rajmahal) }

S_5 : Sure, their phone number is 01223244955 and the postcode is cb58rg. Is there anything else I could help you with?
U_5: That is all I need.

Figure 1: A sample conversation from the MultiWOZ (Budzianowski et al., 2018) dataset (dialogue id PMUL3336).

1 Introduction

In a goal-oriented or task-oriented dialogue system, Dialogue State Tracking (DST) refers to the problem of extracting the goal or intention shown by the user at each turn. The user's goals are captured through a set of dialogue states which are the system's internal representation of the ongoing conversation. DST is essential because it not only helps to understand the user's requirement but also impacts the next dialogue generation. In this era of immersive AI, task-based dialogue systems are gaining popularity day by day. As a result, dealing with a large number of domains and slots will soon become a real problem for task-based chatbots. In this work, we propose a scalable and extensible solution framework for DST to address this forthcoming issue.

We now briefly define DST with an illustration shown in Fig 1. Let U_t and S_t be the user and system utterance respectively at turn t. Then a task-based conversation is generally expressed as $D = \{U_0, (S_1, U_1), \cdots, (S_n, U_n)\}$. Let belief state B_t be the ground-truth dialogue state for turn t. B_t represents the set of (domain, slot, slot-value) triplets that have been extracted so far till turn t. The task of DST is to predict B_t given the dialogue history till turn t.

The solution framework for the DST model can be broadly categorized into three classes - i) picklist-based, ii) generation-based, and iii) end-to-end modeling. The first two methods approach the DST problem explicitly, whereas the third class solves it as a part of end-to-end modeling of the task-based dialogue system. Picklist-based models (Mrkšić et al., 2017; Nouri and Hosseini-Asl, 2018; Zhong et al., 2018; Goel et al., 2019) find the value of a given domain-slot pair from a pre-defined candidate set. This is why these methods need access to the complete ontology of the dataset. This type of modeling can be used only when the candidate set is limited. But in reality, there are many slots (e.g. name, time, etc.) where the range of values can be indefinitely large. Generation-based approaches (Gao et al., 2019; Wu et al., 2019; Kim et al., 2020; Heck et al., 2020) solve this problem by generating the slot-value directly from the dialogue history. These methods usually formulate the slot-value prediction as a reading comprehension (Chen et al., 2017) or text summarization (See et al., 2017) task. There are hybrid models (Zhang et al., 2020) which take the advantages of both picklist and generation-based methods by choosing the slot-value prediction strategy based on the type of slot. On the other hand, end-to-end models (Hosseini-Asl et al., 2020; Wu et al., 2020; Lin et al., 2020; Mehri et al., 2020) aim to unify multiple sub-tasks of a task-oriented dialogue system using a single model. They have the advantage of being fully generative and are usually trained as a conditional or causal language model to generate the next system utterance.

Although recent progress in generation-based and end-to-end approaches has shown significant performance gain in DST, there are still some scalability and extensibility issues that need to be addressed. These issues mainly occur due to two properties - i) predicting domain and slot as a pair, ii) dependency of model parameters on number domains and slots. All the existing DST solutions hold either of these properties and in most cases both. The first property leads to $O(|S|)$ belief state prediction time for each turn where S is the set of all possible domain-slot pairs in a given dataset. In the worst case, $|S| = |M| \times |N|$ where M and N are the sets of unique domains and slots respectively. Since task-based chatbots are designed to work in real-time, reducing time complexity is of critical need. Ren et al. (2019) tackles this issue by predicting domain and slot sequentially and thereby reducing the time complexity to $O(N)$ using their $O(1)$ domain prediction strategy. However, their domain prediction depends on the ordering of domains which can be hard to maintain in a real setup. They also satisfy the second property due to the inclusion of the previous belief state as input. Even though this kind of auxiliary feature has been helpful in improving the joint accuracy (Kim et al., 2020; Heck et al., 2020), it makes the model difficult to extend. The end-to-end models also possess the second property because they encode the previous belief state along with dialogue history to represent a complete turn (Hosseini-Asl et al., 2020). With the growing popularity of task-based conversational systems, we can anticipate larger datasets with lots of domains and slots to be used in the future for the training and development of such systems. Since these datasets will contain a large set of unique domains and slots, scalability and extensibility will become an issue for the existing models.

In this paper, we propose a Hierarchical DST (Hi-DST) model to tackle the issues discussed above. We break the DST task into a hierarchy of four generic sub-tasks - domain change prediction, domain prediction, slot action prediction, and slot-value prediction. We adopt the triple copy strategy (Heck et al., 2020) for slot-value prediction and use the neural span-based question-answering method to extract the slot values from the utterances directly. In contrast to others, we reduce the problem of slot-value prediction to SQuAD (Rajpurkar et al., 2016) to leverage transfer learning. We keep our model parameters independent of the number of domains/slots. This is why we refrain from using any kind of auxiliary features that depend on the domain/slot set. Contributions of our work can be summarized as follows- [1]

- We present Hi-DST, a scalable and extensible DST solution that adopts hierarchical modeling without any dependency on the number of domains and slots. Hi-DST achieves $O(|M| + |N|$ belief state prediction for each turn where M and N are the sets of unique domain and slot respectively.

- We show that Hi-DST achieves a comparable performance to existing DST models while being scalable and extensible simultaneously.

[1]Code is available at github.com/SuvodipDey/Hi-DST

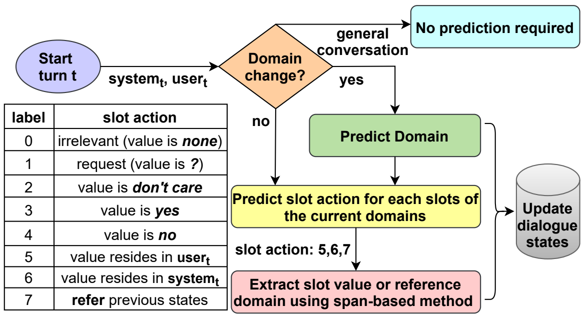
- We argue that the hierarchical structure helps in the explainability of the model and makes it easily extensible to new datasets with a much larger number of domains and slots.

2 Hierarchical DST (Hi-DST)

The core idea behind our approach is to decouple the prediction of domain-slot pairs to achieve belief state prediction in $O(|M| + |N|)$ time. We also keep our model free from any kind of dependency on the number of domains and slots to make it easily extensible. We propose Hi-DST that comprises of four generic components: domain change prediction (section 2.1), domain prediction (section 2.2), slot-action prediction (section 2.3), and slot-value prediction (section 2.4). During prediction (section 2.5), we first detect any change in domain. If there is a change in domain predicted, we run domain prediction and update the set of current domain(s) that keeps track of the active domains for a given turn. We next predict the appropriate actions necessary for relevant domain-slot pairs. Finally, we extract the slot values using span-based method (Chen et al., 2017) when required. We incrementally update our predicted dialogue states at each turn to get the desired belief state. Fig. 2 shows the workflow of our proposed approach.

2.1 Domain Change Prediction

In a task-based conversation, a user can converse about multiple domains and switch between them if necessary. The objective of this component is to detect the point of domain changes. We formulate it as a ternary classification problem. A prediction of 0 represents that there is no change in domain. In this case, we use the domain set of the previous



label	slot action
0	irrelevant (value is ***none***)
1	request (value is ***?***)
2	value is ***don't care***
3	value is ***yes***
4	value is ***no***
5	value resides in **user$_t$**
6	value resides in **system$_t$**
7	**refer** previous states

Figure 2: Workflow of proposed DST model. System and user utterance of turn t are represented as $system_t$ and $user_t$ respectively. The figure shows only the general slot actions.

turn as current domains. Prediction 1 indicates a domain change in the current turn. Here, we need to run the domain prediction model to get the new domains. Finally, class label 2 represents a general conversation (like greeting, thanking, etc.). In this case, we do no further prediction as the user is not showing any additional intention. Basically, this model component captures the theme of a dialogue turn in an abstract way and guides the subsequent predictions accordingly.

We model this three-class classification problem using BERT (Devlin et al., 2019) finetuning. Let S_t and U_t be the system and user utterances at turn t. Then the objective of this model is to find the probability of $p(y|S_t, U_t)$ where $y \in \{0, 1, 2\}$. Let $X_t \in \mathbb{R}^d$ be the encoding of utterance pair (S_t, U_t) where d=768 be the dimension of the BERT embedding. We compute X_t by taking an average of the token embeddings of BERT's second-last hidden layer with ([CLS]S_t[SEP]U_t) as input. We pass X_t through a linear layer of dimension ($d \times$ 3) to find the class probabilities. We use a cross-entropy loss to update the model parameters.

2.2 Domain Prediction

The objective of this component is to find the set of relevant domains in a given user turn. We use a binary classification model to predict 1 if a given domain is relevant and 0 otherwise. Let D be the set of unique domains. Then the goal of the domain prediction model is to find the probability of $p(y|S_t, U_t, d_j)$ where $y \in \{0, 1\}$, and $d_j \in D$. We run this prediction for each domain to obtain the set of current domains.

We encode a specific domain using pre-trained GloVe (Pennington et al., 2014) embedding of dimension d_1 followed by a linear layer and GeLu (Hendrycks and Gimpel, 2020) activation. Let Z be the encoding of domain d_j. So, $Z = GeLu(l_1(Glove(d_j))) \in \mathbb{R}^{d_2}$ where l_1 is a linear layer of dimension of $(d_1 \times d_2)$.

Next, we encode the utterances using BERT. Let G_t be the token representation of utterance pair (S_t, U_t) generated by BERT tokenization. Let $H_t \in \mathbb{R}^{d_2 \times L}$ be the output of BERT's second-to-last hidden layer with input G_t where L is the maximum sequence length and $d_2 = 768$ is the dimension of the BERT embedding. To put attention on relevant tokens, we take a linear combination of the column-vectors of H_t using scaled dot-product attention (Vaswani et al., 2017). We express our final utterance encoding as $X_t = \sum_{l=1}^{L} \alpha_l H_{tl}$ where $X_t \in \mathbb{R}^{d_2}$, $H_{tl} \in \mathbb{R}^{d_2}$ is the output vector of the l^{th} token, and attention $\alpha = softmax(H_t^T Z / \sqrt{d_2}) \in \mathbb{R}^L$. We now concatenate X_t and Z and pass it through a linear classification head of dimension $(2d_2 \times 2)$ to find the class probabilities. We use a softmax classifier with cross-entropy loss. We do not update the GloVe embeddings of the domains during back propagation to extend the model easily for unseen domains.

2.3 Slot Action Prediction

In this component, we find the relevant slots from the predicted set of domains for a given turn. We achieve this by a slot action model that predicts suitable action for a given domain-slot pair. Let D be the set of current domains at turn t. Let C_i be the set of slots in the domain d_i and A be the set of actions. Then the objective of this model is to find the probability of $p(y|S_t, U_t, c_{ij}, d_i)\ \forall i, j$ where $y \in A$, $c_{ij} \in C_i$, and $d_i \in D$.

Based on our analysis, we define eight general and two dataset-speficic actions described in Table 1. Slot-action 0 (NONE) indicates that a domain-slot pair is irrelevant. All the slot-actions between 1 and 4 indicate that the slot-value needs to be inferred because it cannot be extracted directly from the utterances. Slot action 5 (EXT$_{usr}$) represents that the slot-value resides in the *current user utterance* U_t. Slot action 6 (EXT$_{sys}$) indicates that the slot-value is *informed/recommended by the system* and can be extracted from the current system utterance S_t. Finally, slot action 7 (REF) means that the slot-value is *referred to some previous slot-value in the belief state*. Besides the general actions, we have two non-trivial slot-actions specific to Multi-WOZ dataset. The first one is HTL$_{type}$ for *(hotel, type, hotel)* triplet. We add this action because the annotation for this triplet is inconsistent throughout the dataset (Wu et al., 2019). The second one is PPL$_1$ for triplet *(d, people, 1)* for any domain d. This triplet often needs to be inferred rather than extracted directly as shown in the example in Table 1. We found that it is better to handle such dataset-specific non-trivial cases with a new slot action since these values are difficult to extract using span-based approaches.

Our slot action prediction model is very similar to the domain prediction model of Section 2.2. Instead of a domain, here we encode a domain-slot pair in a similar fashion. Here, the encoding of a given domain-slot pair (d, c) can be expressed as $Z = GeLu(l_1([Glove(c); Glove(d)]))$ where $Z \in \mathbb{R}^{d_2}$ and l_1 is a linear layer of dimension of $(2d_1 \times d_2)$. The rest of the modeling remains the same as the domain prediction model except for the final classification head. The dimension of the final linear layer becomes $(2d_2 \times k)$ where k is the number of slot actions. GloVe embedding of the domains or slots is not updated during training just like our domain prediction model.

Label	Action	Description	Example
0	NONE	slot is irrelevant, slot-value is *"None"*	In *"I want an expensive place to stay in the west side."*, slots like *Name* and *Parking* are irrelevant.
1	REQ	slot is requested by the user, slot-value is *"?"*	In "*What is their address and phone number?*", use has requested *Address* and *Phone*.
2	DNC	user doesn't care about the slot, slot-value is *"don't care"*	In *"I'm looking for a hotel in the west, internet is optional"*, slot-value for *Internet* will be *"don't care"*.
3	YES	slot-value is *"Yes"*	In *"I need free parking"*, slot-value for *Parking* is *"Yes"*.
4	NO	slot-value is *"No"*	In *"I don't need internet or free parking"*, slot-value for *Internet* and *Parking* is *"No"*.
5	EXT$_{usr}$	slot-value needs to be be extracted from the current user utterance	S_t : Okay, where would you like to depart from? U_t: I'd like to leave from **Cambridge**, please.
6	EXT$_{sys}$	slot-value needs to be be extracted from the current system utterance	S_t: I recommend **Kettle's Yard** on Castle Street which is a museum. U_t: Could I get the postcode for that museum?
7	REF	the value of the slot needs to be be referred	In *"I'd like to go see a college that's in the **same area as the hotel**"*, slot-value of *Area* refers to a previously extracted value.
8	HTL$_{type}$	type of the hotel is "hotel"	*"I also need to find a 2 star room ."*
9	PPL$_1$	number of people is 1	S_t : How many tickets would you like? U_t : Just for **myself** , please.

Table 1: Description of slot actions with example.

2.4 Slot Value Prediction

The fourth and final component of Hi-DST is the slot-value prediction for a given domain-slot pair. We need slot-value prediction model for slot actions 5 (EXT$_{usr}$), 6 (EXT$_{sys}$), and 7 (REF) because for the rest it can be inferred directly. If the predicted slot-action for a given domain-slot pair is 5 and 6, we need to extract the slot-value from the current user and system utterance respectively. Whereas for slot-action 7, we have to find the reference point of the slot-value from the user utterance and then copy its value. This kind of strategy for slot-value prediction is called triple copy strategy (Heck et al., 2020) and has been shown to be beneficial for DST. We reduce these three kinds of slot-value prediction to the span-based question answering problem of the SQuAD dataset (Rajpurkar et al., 2016). By doing so we can directly finetune the span-based neural comprehension model (Chen et al., 2017) pre-trained on SQuAD and reap the benefits of transfer learning. In the SQuAD dataset, the input is a pair of a question and context and the objective is to predict the span (start and end index) of the answer in the given context. We reduce our slot-value prediction problem to SQuAD as follows:

Extract from User Utterance (EXT$_{usr}$): For slot action 5 (EXT$_{usr}$), the value of a given domain slot pair is present in the current user utterance. So, we set the context to U_t. We generate the question by converting the given domain-slot pair into an English sentence. For example, (hotel-name) becomes *"What is the name of the hotel?"*, (train-destination) becomes *"What is the destination of the train?"*, and so on. The motivation for such question generation is to match the format of SQuAD. In this work, we use rule-based question generation like DS-DST (Zhang et al., 2020) as the set of domain-slot pairs is limited. It would be nice to have a model-based approach to handle question generation on a large scale.

Extract from System utterance (EXT$_{sys}$): In this scenario, the value of a given domain slot pair is present in the current system utterance. It occurs when the user accepts the system's recommendation/suggestion. The reduction is absolutely similar to the earlier case except the context now being the current system utterance S_t. If the set of informed slots by the system at each turn is available, then we do not need to extract the slot value. Instead, we can copy the slot-value of the domain-slot pair directly from that set during prediction.

Refer (REF): In this case, the slot-value for a given domain-slot pair refers to a previously extracted value. Hence, our objective here is to find the appropriate reference point in the belief state of the previous turn and then copy its value. Let the reference point for a given domain slot pair (d, s) be (d^{ref}, s^{ref}). In general, we observe that slots s and s^{ref} remain the same. So, the main challenge is to find the reference domain d^{ref}. We formulate the problem of finding the reference domain similar to the formulation of slot action 5 (EXT$_{usr}$) and 6 (EXT$_{sys}$). The context is set to be the current user utterance U_t. We convert a domain-slot pair into a question in a slightly different manner. For example, the REF instance shown in Table 1, we form the question as *"What is the reference point of the attraction area?"* and the model is trained to extract the reference domain *"hotel"*. There are few special cases where the original slot s does not match the reference slot s^{ref}. For instance in the MultiWOZ dataset, slots like *destination* and *departure* refers *name*. In this work, we resolve these slot references manually while creating the training data for this phase, since such examples were limited in number.

2.5 Predictive Algorithm

We now briefly describe our predictive algorithm for a single conversation. Let D be the set of current domain(s) that keeps track of the active domains for a given turn. Let B be the set of predicted belief states. Initially, both D and B are empty. Before moving on to the next turn, B and D are updated based on the predictions made for the current turn. For each user turn t with input (S_t, U_t, D, B), we do the following:

- **Step 1**: Run the domain change prediction model (Section 2.1).
  - If the prediction is a general conversation (Class 2), we make $D = \varnothing$ and skip all subsequent predictions for the current turn.
  - If domain change is detected (Class 1), we go to Step 2.
  - If no change in the domain is predicted (Class 0), we do the following:
    - $*$ If the cardinality of the set of current domains (D) is 1, we directly go to the slot action prediction in Step 3.
    - $*$ Otherwise, we go to Step 2 to update D. It gives the model an extra chance to find

a domain when $D = \varnothing$. Whereas, it helps to remove extraneous domains in case of more than one relevant domain.

- **Step 2**: Run domain prediction model (Section 2.2) to get the set of current domains for turn t.
- **Step 3**: Predict slot action (Section 2.3) for each slots of the current domains. If slot action EXT_{usr}, EXT_{sys} or REF is detected, we go to Step 4. Otherwise, the slot-values are directly inferred and updated in the belief state for turn t.
- **Step 4**: Extract the slot-value or reference domain using span-based question-answering method (Section 2.4) and update the belief state B accordingly.

The main purpose of steps 1 and 2 is to predict the relevant domains that are subsequently used for slot value prediction (wherever necessary). This is required due to our decoupling of the domain and slot predictions. We observe that in Step 4 for slot action REF, the model sometimes fails to find the reference domain. This occurs when the user does not explicitly mention the reference domain. For example, "*Could you please book train tickets for the same group?*". In such cases, we select the most recent domain that contains the reference slot s^{ref} as the reference domain d^{ref}.

3 Dataset and Experimental Setups

3.1 Dataset

We use the MultiWOZ dataset (Budzianowski et al., 2018) for experimentation. It is one of the largest multi-domain conversation corpus available for task-oriented dialogue systems. We perform our experiments on MultiWOZ 2.1 (Eric et al., 2020) and MultiWOZ 2.2 (Zang et al., 2020). Both the datasets are updated versions of the original Multi-WOZ dataset and contain fixes to some noisy annotations. Table 2 and 3 shows some basic statistics of the dataset.

3.2 Evaluation Metric

Dialogue state tracking is broadly evaluated using several metrics like joint accuracy, slot accuracy, and average joint accuracy (Rastogi et al., 2020). The primary metric for DST is joint accuracy or joint goal accuracy. Joint accuracy is defined by the fraction of turns where the predicted belief state exactly matches the ground truth (Wu et al., 2019). In this work, we only use joint accuracy so that we can directly compare Hi-DST with other models.

There are a lot of instances in the MultiWOZ dataset where the labeled slot value for a given domain-slot pair is not present in the dialogues in its exact form. Rather some variant of the slot value exists like *cafe jello* instead of *cafe jello gallery*, *centre* instead of *center*, and so on. This can cause a problem for a fair evaluation of span-based slot value prediction. TripPy (Heck et al., 2020) addresses this issue using a label variant map [2] where each value is mapped to a set of variants. A match is considered if the predicted slot value exactly matches the ground truth or any of its variants. We follow the same to evaluate Hi-DST.

Data	#Dialogues	#Turns	Avg turns per dialogue
Train	8420	56668	6.73
Dev	1000	7374	7.37
Test	999	7368	7.37

Table 2: Data statistics of MultiWOZ 2.1

Domain	Slots	Conversations
attraction	name, type, area	33.47%
hotel	name, type, parking, area, day, stay, internet, people, stars, price	40.1%
restaurant	name, food, area, day, time, people, price	45.48%
taxi	arrive, departure, leave, destination	18.01%
train	arrive, day, leave, destination, departure, people	37.64%

Table 3: Unique domain-slot pairs for which slot-value needs to be extracted in MultiWOZ 2.1.

3.3 Data Preparation

We now summarize the training data generation for Hi-DST. We use the turn-level belief state rather than the cumulative one in our training process. Let B_t be the set of belief state at turn t. Then $T_t = B_t \setminus B_{t-1}$ be the turn-level belief state for turn t. We ignore the turns for data preparation where $T_t = \varnothing$.

Let D_t be the set of domains in T_t. Then for the domain change component, we compare D_t and D_{t-1}. If there is no change, we label 0, and 1 otherwise. Annotation for general conversation is available in the MultiWOZ dataset. If this annotation is not available in a dataset, we can ignore this class and train the domain change model with only two classes. For the domain model, we label

[2] gitlab.cs.uni-duesseldorf.de/general/dsml/trippy-public/blob/master/dataset_config/multiwoz21.json

Data	Metric	domain change model			domain model		slot action model									
		0	1	2	0	1	0	1	2	3	4	5	6	7	8	9
Train	**Precision**	0.96	0.94	1.0	0.99	0.98	0.98	0.98	0.79	0.95	0.61	0.98	0.88	0.84	0.90	0.81
	Recall	0.99	0.85	0.97	1.0	0.96	0.98	0.97	0.81	0.95	0.60	0.99	0.74	0.84	0.55	0.92
	F1-score	0.97	0.89	0.98	0.99	0.97	0.98	0.98	0.80	0.95	0.60	0.98	0.80	0.84	0.69	0.86
	Support	31060	7377	9879	90486	16999	136797	12813	1942	3005	203	52238	5264	2747	377	1450
Dev	**Precision**	0.95	0.91	0.99	0.99	0.95	0.97	0.96	0.62	0.92	0.59	0.97	0.75	0.78	0.68	0.74
	Recall	0.98	0.85	0.97	0.99	0.94	0.97	0.96	0.71	0.88	0.71	0.98	0.59	0.77	0.37	0.84
	F1-score	0.97	0.88	0.98	0.99	0.95	0.97	0.96	0.66	0.90	0.65	0.98	0.66	0.78	0.48	0.79
	Support	4052	1065	1249	11955	2227	18206	1691	160	366	14	7214	598	356	57	143
Test	**Precision**	0.95	0.91	0.99	0.99	0.96	0.96	0.96	0.75	0.90	0.27	0.96	0.82	0.80	0.76	0.80
	Recall	0.98	0.83	0.96	0.99	0.93	0.97	0.97	0.69	0.89	0.36	0.98	0.51	0.78	0.48	0.84
	F1-score	0.96	0.87	0.98	0.99	0.94	0.97	0.96	0.72	0.89	0.31	0.97	0.63	0.79	0.59	0.82
	Support	4059	1078	1235	11949	2289	18646	1803	236	362	11	7168	794	359	71	170

Table 4: Class-wise performance of *domain change*, *domain*, and *slot action* models on MultiWOZ 2.1 dataset.

Data	Accuracy	Support
Train	0.983	137,185
Dev	0.979	18,293
Test	0.979	18,551

Table 5: Individual performance of slot-value prediction model on MultiWOZ 2.1 dataset.

a domain d as 1 if $d \in D_t$ and 0 otherwise.

Let C_t be the set of domain-slot pairs in T_t. We use C_t to generate the labels for slot action as described in Table 1. We take the help of the span index annotation in MultiWOZ for generating the data for the slot-value model. We also added negative samples for irrelevant domain-slot pairs for which the start and end index is set to 0.

3.4 Training Details

We implemented our models using PyTorch and Huggingface (Wolf et al., 2020) libraries in Python 3.7. All the experiments were performed on an Nvidia Tesla P100 machine with 16GB of memory. We used AdamW (Loshchilov and Hutter, 2019) optimizer and set the learning rate and adam's epsilon value to 2e-5 and 1e-8 respectively. We trained all the models for 4 epochs and chose the final model having minimum validation loss.

We used a common configuration for the domain change, domain, and slot action model. We trained these three models using pre-trained *bert-base-uncased* model. Besides the BERT model, we applied a drop out of 0.3 to the input of the final classification head. We also used gradient norm clipping with a maximum threshold of 2. The maximum token length (L) was set to 200. Individual model performances are shown in Table 4.

We finetuned *bert-large-uncased-whole-word-masking-finetuned-squad* [3] model for our span-based slot-value prediction model. The maximum token length (L) was set to 100. We evaluate the slot-value prediction model using accuracy i.e the fraction of data where the predicted span exactly matches the ground-truth. The individual performance is shown in Table 5.

4 Result and Analysis

4.1 Result

We report our result on MultiWOZ 2.1 (Eric et al., 2020) and MultiWOZ 2.2 (Zang et al., 2020) dataset in Table 6. We compare our models with SGD baseline (Rastogi et al., 2020), TRADE (Wu et al., 2019), DS-DST (Zhang et al., 2020), TripPy (Heck et al., 2020), and simple-TOD (Hosseini-Asl et al., 2020). Although the performance of Hi-DST is comparable to existing models, its performance is generally lower than most of the recent models. We discuss this point in Section 4.2.4 elaborately while analyzing our method.

4.2 Analysis

We analyse Hi-DST in four different aspects: scalability (Section 4.2.1), extensibility (Section 4.2.2), explainability (Section 4.2.3), and performance analysis (Section 4.2.4).

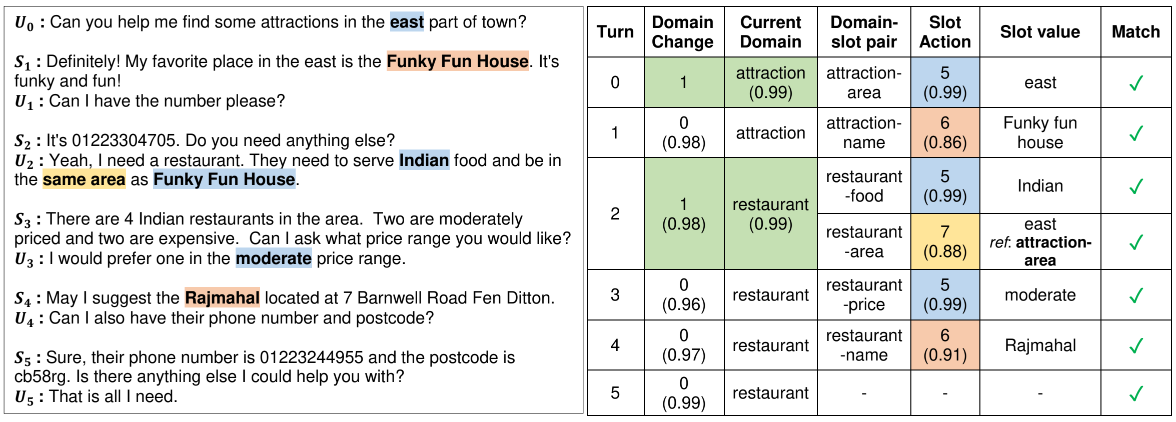
4.2.1 Scalability

We made Hi-DST scalable by ensuring two things: i) making slot-value prediction completely rely on span-based QA, ii) decoupling the prediction of domain and slot. As a result of this kind of modeling, Hi-DST takes $O(|M| + |N|)$ time to predict a belief state for a given user turn where M and

DST Model	MultiWOZ 2.1	MultiWOZ 2.2
SGD baseline	43.4%	42.0%
TRADE	46.0%	45.4%
TripPy (without auxiliary features)	49.23%	-
DS-DST	51.2%	51.7%
TripPy	55.3%	-
SimpleTOD	56.45%	-
Hi-DST (Ours)	49.16%	49.44%

Table 6: Joint accuracy comparison

[3] huggingface.co/transformers/pretrained_models.html



U_0 : Can you help me find some attractions in the **east** part of town?

S_1 : Definitely! My favorite place in the east is the **Funky Fun House**. It's funky and fun!
U_1 : Can I have the number please?

S_2 : It's 01223304705. Do you need anything else?
U_2 : Yeah, I need a restaurant. They need to serve **Indian** food and be in the **same area** as **Funky Fun House**.

S_3 : There are 4 Indian restaurants in the area. Two are moderately priced and two are expensive. Can I ask what price range you would like?
U_3 : I would prefer one in the **moderate** price range.

S_4 : May I suggest the **Rajmahal** located at 7 Barnwell Road Fen Ditton.
U_4 : Can I also have their phone number and postcode?

S_5 : Sure, their phone number is 01223244955 and the postcode is cb58rg. Is there anything else I could help you with?
U_5 : That is all I need.

Turn	Domain Change	Current Domain	Domain-slot pair	Slot Action	Slot value	Match
0	1	attraction (0.99)	attraction-area	5 (0.99)	east	✓
1	0 (0.98)	attraction	attraction-name	6 (0.86)	Funky fun house	✓
2	1 (0.98)	restaurant (0.99)	restaurant-food	5 (0.99)	Indian	✓
			restaurant-area	7 (0.88)	east *ref:* **attraction-area**	✓
3	0 (0.96)	restaurant	restaurant-price	5 (0.99)	moderate	✓
4	0 (0.97)	restaurant	restaurant-name	6 (0.91)	Rajmahal	✓
5	0 (0.99)	restaurant	-	-	-	✓

Figure 3: Illustration of the working of Hi-DST.

N are the sets of unique domains and slots respectively. It is to be noted that all turns do not require $O(|M|+|N|)$. It is true only for those turns where we need to update the set of current domains. Belief state prediction can take $O(|N|)$ and $O(1)$ when domain change prediction is 0 and 2 respectively. Since the number of domain changes and general dialogues in a task-based conversation is very limited, the dominating factor is $O(|N|)$. To the best of our knowledge, $O(|M|+|N|)$ is the best time complexity for dialogue state prediction without any kind of dependency on auxiliary features and domain statistics.

4.2.2 Extensibility

All four components of Hi-DST are completely independent of the number of domains and slots. So, the number of model parameters will remain the same for any dataset. This is why Hi-DST is easily extensible to datasets with a large number of domains/slots. Moreover, we convert a domain/slot using Glove embedding which is not updated during training. This property enables the model to be used in zero-shot and few-shot scenarios.

4.2.3 Explainability

As described earlier, we break the DST task into a hierarchy of generic sub-tasks. Due to this hierarchical structure, we can look at Hi-DST as a series of meaningful actions which closely resemble human-like decision-making. In Fig 3, we show the details of a Hi-DST prediction along with the confidence of each decision. Firstly, we can observe that it is human-readable and self-explanatory. Secondly, the probability score quantifies each decision and helps in debuggability. For a wrong prediction, we can easily eyeball the probability scores and find the root cause of the mistake. Thirdly, the model is capable of detecting user requests which enable the understanding of complete user intent.

4.2.4 Performance Analysis

We now do a critical analysis of our model performance. Even though having some good properties (like scalability, extensibility, and explainability), our accuracy is lower in comparison to the state-of-the-art models. There are several factors that limit the performance of Hi-DST. Firstly, most of the high-performing models don't predict domain. We introduce an extra uncertainty in our model through domain prediction which reduces the accuracy but helps in scalability. Secondly, there are a lot of wrong and inconsistent annotations in the MultiWOZ dataset (Zang et al., 2020). These noisy annotations can have a big impact on the predictions since our model components are trained independently. Thirdly, the higher accuracy models use auxiliary features to preserve contextual information. For example, the inclusion of the previous belief state has been shown to be beneficial (Heck et al., 2020) to improve accuracy. These features can also help to adapt to the inconsistencies in the data. We can observe in Table 6 that the joint accuracy of TripPy drops from 55.3% to 49.2% without the auxiliary features which is very similar to the performance of our model. Although these auxiliary features are helpful, they are dependent on the number of domains and slots which makes them difficult to extend to a new dataset with different domains and slots. Whereas, due to the generic modules and the extensible nature of Hi-DST, we can easily adapt to a new dataset. We also do not need to re-train or finetune all the components for a new dataset. For example, if a newer dataset has the same set of domains, fine-tuning the slot-action model is enough to get a decent result.

5 Conclusion

In this work, we propose Hierarchical-DST (Hi-DST), a scalable and extensible solution framework for DST. We split the task of DST into four generic modules that not only make Hi-DST scalable and extensible for larger datasets but also improve its explainability. Hi-DST takes $O(|M| + |N|)$ time belief state prediction per user turn and achieves comparable performance to existing DST models. We discuss the performance trade-off due to the enforcement of scalability and extensibility. As future work, we want to continue our experimentation in zero-shot and few-shot scenarios and investigate the efficiency of Hi-DST in complex datasets like SGD (Rastogi et al., 2020). We would also like to explore the possibility of including additional information or auxiliary features without impacting the desirable properties of Hi-DST such as scalability, explainability etc.

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 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, Brussels, Belgium. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

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, Marseille, France. European Language Resources Association.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273, Stockholm, Sweden. Association for Computational Linguistics.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. HyST: A Hybrid Approach for Flexible and Accurate Dialogue State Tracking. In *Proc. Interspeech 2019*, pages 1458–1462.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. TripPy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2020. Gaussian error linear units (gelus).

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020. Efficient dialogue state tracking by selectively overwriting memory. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582, Online. Association for Computational Linguistics.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. MinTL: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Shikib Mehri, Mihail Eric, and Dilek Hakkani-Tur. 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue.

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, Vancouver, Canada. Association for Computational Linguistics.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking. In *NeurIPS 2018, 2nd Conversational AI workshop*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:8689–8696.

Liliang Ren, Jianmo Ni, and Julian J. McAuley. 2019. Scalable and accurate dialogue state tracking via hierarchical sequence generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 1876–1885. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929, Online. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

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.

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

Victor Zhong, Caiming Xiong, and Richard Socher. 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)*, pages 1458–1467, Melbourne, Australia. Association for Computational Linguistics.