Similarity-based Multi-Domain Dialogue State Tracking with Copy Mechanisms for Task-based Virtual Personal Assistants

Jarana Manotumruksa
University College London
jarana.manotumruksa@gmail.com

Jeffrey Dalton
University of Glasgow
jeff.dalton@glasgow.ac.uk

Edgar Meij
Bloomberg
emeij@bloomberg.net

Emine Yilmaz
University College London, Amazon
emine.yilmaz@ucl.ac.uk

ABSTRACT

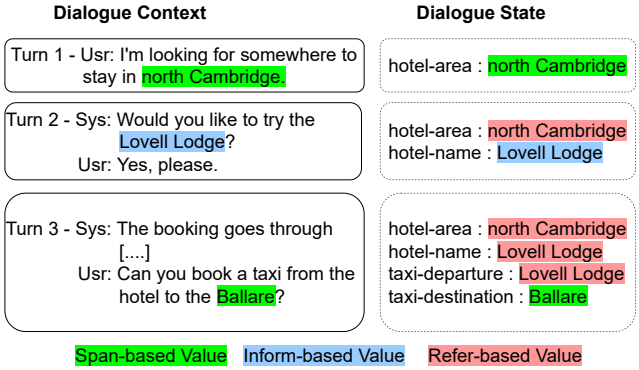
Task-based Virtual Personal Assistants (VPAs) rely on multi-domain Dialogue State Tracking (DST) models to monitor goals throughout a conversation. Previously proposed models show promising results on established benchmarks, but they have difficulty adapting to unseen domains due to domain-specific parameters in their model architectures. We propose a new Similarity-based Multi-domain Dialogue State Tracking model (SM-DST) that uses retrieval-inspired and fine-grained contextual token-level similarity approaches to efficiently and effectively track dialogue state. The key difference with state-of-the-art DST models is that SM-DST has a single model with shared parameters across domains and slots. Because we base SM-DST on similarity it allows the transfer of tracking information between semantically related domains as well as to unseen domains without retraining. Furthermore, we leverage copy mechanisms that consider the system's response and the dialogue state from previous turn predictions, allowing it to more effectively track dialogue state for complex conversations. We evaluate SM-DST on three variants of the MultiWOZ DST benchmark datasets. The results demonstrate that SM-DST significantly and consistently outperforms state-of-the-art models across all datasets by absolute 5-18% and 3-25% in the few- and zero-shot settings, respectively.

ACM Reference Format:
Jarana Manotumruksa, Jeffrey Dalton, Edgar Meij, and Emine Yilmaz. 2022. Similarity-based Multi-Domain Dialogue State Tracking with Copy Mechanisms for Task-based Virtual Personal Assistants. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 INTRODUCTION

Task-based Virtual Personal Assistants (VPAs) interact with users in natural language to complete tasks. Modern VPAs such as Google Assistant, Siri and Alexa support complex tasks that span multiple domains. Figure 1 illustrates a multi-domain conversation with requests to book a hotel and a taxi ride to pick them up. Keeping track of the state across these sub-tasks (domains) is called Dialogue State Tracking (DST) and is an essential component that extracts the intents expressed during the conversation and encodes them in a set of states (i.e. a set of domain-slot pairs and their values)[5, 18]. DST models usually rely on an ontology that defines 1) slots for a particular task (e.g. departure and destination for taxi booking) and 2) the set of possible values for each. A DST model takes the current utterance and previous dialogue history (with its state) and predicts 1) if it mentions a given slot and 2) if so, its value. Based on the current dialogue state VPAs decide the next optimal action (e.g. a database query or a natural language generation response).

Figure 1: An example multi-domain dialogue; green and red terms represent correctly and incorrectly predicted states.



Current DST models can be categorised into ontology-based and span-based. Ontology-based models [15, 23, 24] require full access to a pre-defined ontology, where all domain-slot pairs and their values are known in advance. They simplify the task of DST by performing classification over the candidate values for a given domain-slot pair and dialogue context. This works well for domains with fixed categorical values, but their main disadvantage is that knowing all possible values for open-domain slots such as a place name is difficult or impossible. As a result, such models suffer from the out-of-vocabulary problem for unseen values [19]. To address this, span-based DST models [1, 7, 11, 13, 19] extract slot values directly from the utterances and dialogue history without requiring a pre-defined list of candidate values. These models assume that the intent is expressed within the utterances and can be selected. However, this also suffers from the issue that values may be implied.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France.

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/XXXXXX.XXXXXX

We show an example in Figure 1, where in Turn 2 the user implicitly provides the value of "hotel-name" by selecting the value informed by the system. In Turn 3, the user implicitly provides the value of "taxi-departure" by referring to the value of "hotel-name" provided in the previous turn. To address the challenges of coreference and value sharing, several recent span-based DST models [7, 11] use copy mechanisms to refer to a slot's value from the dialogue state. Despite the fact that current ontology and span-based models show promising results on standard multi-domain datasets[5, 22], their architectures are limited in the number of domains/slots and usually required labeled data (supervised or weakly supervised) in order to adapt to new domains and unseen slots.

We propose a new DST approach inspired by retrieval and notions of similarity. We argue that DST models should have one model with shared parameters and use slot and domain similarity. As a result, the proposed model be able to adapt to new domains over an increasingly diverse area of application. The proposed model leverages the similarity between slots across different domains with similar values (e.g. the "area" slot for the hotel, restaurant and attraction domains). Similar domains are likely to share similar slots and intents. For example, booking a train vs ordering a taxi, both are transportation and require a departure (pickup point) and destination (drop-off point) slots with similar semantics. A model should be able to transfer by learning from the train domain for the taxi domain without additional training data. This is important because collecting training data for new domains is also challenging, expensive, and time-consuming [9, 14, 18]. The contributions in this work include the following:

- We propose a novel Similarity-based Multi-Domain Dialogue State Tracking model (SM-DST) that uses retrieval-inspired similarity as well as state copy mechanisms to track dialogue state. Key components of the SM-DST model are scalable and transferable because they share their parameters across domains allowing them to track knowledge between similar domains and values for unseen slots
- We propose an effective token-level correlation estimator to estimate the similarity between the current dialogue context and candidate slots. To the best of our knowledge, this work is the first to estimate the similarity between dialogue context and slots at token-level instead of sentence-level.
- We perform experiments on multi-domain DST benchmark datasets. Results show the proposed model consistently and significantly outperforms previous baselines in standard supervised, few-shot, and zero-shot settings.

2 RELATED WORK

The diverse DST models proposed in previous literature can be categorised into two main types: the ontology-based [15, 21, 23, 24] and span-based models [1, 2, 7, 11, 13, 16, 19]. Recently, these existing DST models often exploit pre-trained contextual language models (e.g. ELMO [17] and BERT [4]) to encode the utterances, system responses, and dialogue history. They train supervised classification models on these encoded dialogue representations to track the current dialogue state. For example, Zhang et al. [23] propose an ontology-based DST model that exploits BERT to learn a dialogue-slot representation from the utterance and slot description. Then, they use a separate BERT model to encode corresponding values of the slot. Given a set of candidate values for a given slot from the pre-defined ontology their model predicts the dialogue state based on the similarity between the encoded candidate values and dialogue-slot representations. In contrast, Lee et al. [15] exploit BERT for two separate independent encoding of the dialogue context and domain-slot pairs. They use attention models to learn dialogue-slot representation and scores for every candidate slot-value pair in a non-parametric manner using a distance measure.

Recent works[1, 2, 7, 11] focus on the span-based approach to address the out-of-vocabulary problem for unseen-slot values as well as generalisation issues of previously proposed ontology-based DST models. Chao and Lane [1] propose a span-based model built on BERT. They replace the original language-modelling head with two specific heads: one that performs per-slot utterance level classification to predict whether a given slot is active or mentioned in a given utterance and another per-slot head that predicts the beginning and the end of the span that represents a value of the slot. Chen et al. [2] propose a schema-guided model that exploits graph attention networks to learn the hidden slot-slot relations from the pre-defined schema and predicts the dialogue state from utterances and learned schema graphs. However, the models proposed in [1, 2] are not efficient because they must predict the dialogue state at every previous turn from scratch. To address this, Kim et al. [11] propose a copy mechanism that incorporates the dialogue state predicted from previous turns when tracking the state at the current turn. Their copy mechanism decides whether a value of a given slot at the current turn can be copied from the previously predicted value of the history. If not, they use separate recurrent neural networks to decode the slot values given the dialogue-slot representation. Inspired by [11], Heck et al. [7] recently propose a triple copy mechanism that can copy a slot value from 1) span prediction component 2) a system response memory that keeps track of the system's response operations, and 3) predicted values from previous turns.

Although existing ontology and span-based DST models show promising results on multiple DST benchmark datasets, their architectures are not scalable because they have per-slot parameters and need to be retrained to learn them for new domains and unseen slots. Several scalable models propose address this issue [13, 15, 19]. Wu et al. [19] propose a transferable span-based DST model that transfers tracking knowledge across different domains and slots. Their model consists of the dialogue encoder, a slot gate, and a state generator, which shared across domains. Kumar et al. [13] extend the scalable DST model proposed by Wu et al. [19] by improving the encoding of dialogue context and slot semantics to robustly capture important dependencies between slots and the conversation history. They use cross-attention to model relationships between the context dialogue and domain-slot pairs at different semantic levels and self-attention to address the coreference and value sharing issue. These scalable DST models can be used in a zero-shot setting for new domains or unseen slot values. However, the existing scalable DST models cannot leverage the auxiliary information such as the system inform memory and the predicted dialogue state from previous turns. Therefore, they may not be able to handle the challenge of coreference and value sharing.

3 SIMILARITY-BASED MULTI-DOMAIN DST WITH COPY MECHANISMS

3.1 Problem Statement and Notation

The goal of Dialogue State Tracking is to record the structured state of the task at a particular turn given the current utterance, and previous system response as well as auxiliary features. Let $\mathcal{DS} = \{DS_1, DS_2, ..., DS_t\}$ be the dialogue state of the user for each turn. Each dialogue state DS_t is a set of tuples (s, v), where $s \in \mathcal{S}$ is a domain-slot pair and v is a value associated with the domain-slot s. Let $\mathcal{X} = \{(U_1, R_1), (U_2, R_2), ..., (U_t, R_t)\}$ be the sequence of user utterance U and system response R pairs at each turn. Each (U_t, R_t) pair can talk about single or multiple domains (e.g. restaurant and taxi) and a certain number of slots (e.g. restaurant-name and taxi-destination) associated with the respective domains. There are two types of auxiliary features: system responses (inform acts) and dialogue state memories. The system inform memory $m^{if}_{s,t} \in \{0, 1\}$ indicates whether a value of slot s is informed by the system in turn t and the dialogue state memory $m^{ds}_{s,t-1} \in \{0, 1\}$ indicates whether slot s's value is predicted by the DST model in the previous turn.

3.2 The architecture of SM-DST

The architecture of SM-DST model is shown in the right part of Figure 2 and consists of four main layers: input, embedding, correlation estimation, and prediction layers. Starting at the bottom of the figure, the input to SM-DST (yellow pentagons) are the current dialogue context $c_t = [U_t, R_t, H_t]$ at turn t, a given domain-slot pair s and auxiliary features including the system inform memory $m^{if}_{s,t} \in \{0, 1\}$ and the dialogue state from the previous turn, $m^{ds}_{s,t-1} \in \{0, 1\}$. $m^{if}_{s,t} = 1$ indicates that the system has recently informed the value of slot s to the user at current turn t, while $m^{ds}_{s,t-1} = 1$ indicates that the value of slot s was predicted in the previous turn $t - 1$. Let $H_t = (U_{t-1}, R_{t-1}), ..., (U_1, R_1)$ be the dialogue history before turn t. Next, the embedding layer of SM-DST consists of the slot encoder and dialogue encoder that encodes the given slot s and dialogue context c_t into separate contextualised representations. The output of the embedding layer is passed to the correlation estimation layer to learn the correlation between the current dialogue context c_t and slot s. Finally, the output from the correlation estimation and embedding layers are passed to the prediction layer that predicts the dialogue state at turn t. In the prediction layer, there are three types of predictors: slot operation, span-based value, and refer-based value.

The architecture of SM-DST differs from the architecture of existing, less scalable DST models (e.g. [1, 7]) in several aspects, illustrated in Figure 2. First, other DST models usually have only one encoder to encode both dialogue context and domain-slot pairs, while SM-DST has two separate encoders to independently encode the dialogue context and domain-slot pairs. Second, other DST models require N slot operation predictors and N span-based value predictors, therefore, the complexity of these models increases as the number of domain-slot pairs increases. Another disadvantage is that they have N predictors and training data for every domain in order to train each predictor. As a result, they are less likely to be able to transfer the tracking knowledge from the seen domains to the unseen domains. In contrast, SM-DST is scalable as it has only three predictors that are shared across different domains. The main advantage of shared predictors is that SM-DST can extract the user's dialogue state for unseen domain-slot pairs even though training data for new domains is not available. The complexity of SM-DST is constant since the parameters of SM-DST do not increase as the number of domain-slot pairs increases. Unlike SM-DST, the existing scalable models (e.g. [13, 19]) cannot leverage the system inform memory and the predicted dialogue state from previous turns.

3.3 The Dialogue and Slot Encoders

The SM-DST model consists of the dialogue encoder and the slot encoder where the former aims to capture the user's intent from the dialogue context and the latter learns the representation of different domain-slot pairs (see the green and red rectangles in Figure 2). The input of the dialogue encoder is the dialogue context c at turn t that consists of current utterance U_t, system response R_t and dialogue history $H_t = (U_{t-1}, R_{t-1}), ..., (U_1, R_1)$. We exploit pre-trained language models[1] (e.g. BERT[4]) as the encoder and the domain-slot encoder to encode their inputs as follows:

$$E_{c,t} = BERT([CLS] \oplus U_t \oplus [SEP] \oplus R_t \oplus [SEP] \oplus H_t \oplus [SEP]), \quad (1)$$

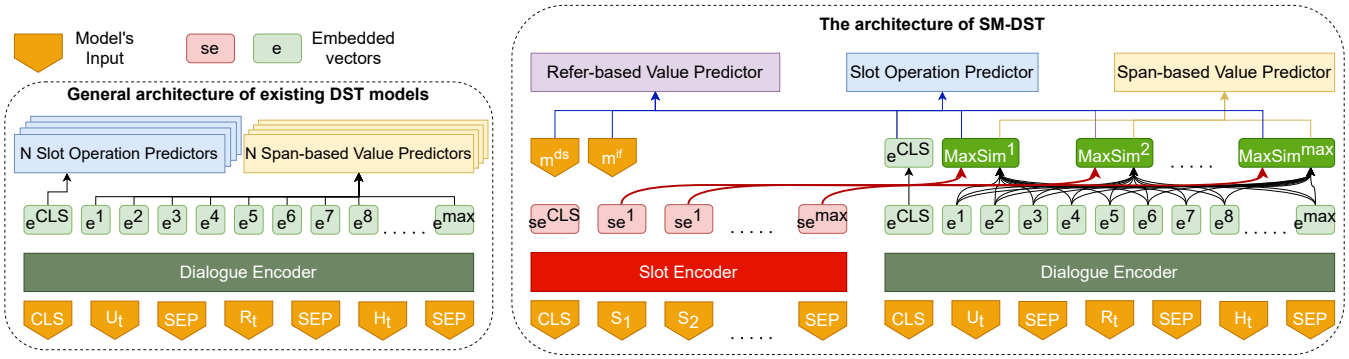
where $\oplus$ is the concatenation operation, $[CLS]$ and $[SEP]$ are BERT's special tokens. $E_{c,t} = [e^{CLS}_t, e^1_t, ..., e^{seq^c_{max}}_t]$ is the output of the dialogue encoder that represents each token in the dialogue context c and seq^c_{max} is the maximum length of the dialogue token sequence. In particular, $e^{CLS}_t \in \mathcal{R}^n$ is the aggregated representation of the total seq^c_{max} sequential input tokens that captures the user's intention from the whole dialogue context c, where n is the size of BERT's contextual embedding dimension. $T_{c,t} = [e^1_t, e^2_t, ..., e^{seq^c_{max}}_t]$ are token-level representation of each token. Given the candidate domain-slot pair s, we use the slot encoder to generate several representations: namely slot, inform-slot and refer-slot representations. Following [15, 19], we exploit a separate pre-trained language model to represent each candidate slot as follows:

$$\begin{aligned} E_s &= BERT([CLS] \oplus desc(s) \oplus [SEP]), \\ E^{if}_s &= BERT([CLS] \oplus [INFORM] \oplus [SEP] \oplus desc(s) \oplus [SEP]), \\ E^{ds}_s &= BERT([CLS] \oplus [REFER] \oplus [SEP] \oplus desc(s) \oplus [SEP]), \end{aligned} \quad (2)$$

where $desc(s)$ is a function that returns a textual description of domain-slot pair s. $[INFORM]$ and $[REFER]$ are special tokens that are used to generate inform- and refer-slot representations. $E_s = [se^{CLS}, se^1, ..., se^{seq^s_{max}}]$ is the output of the slot encoder that represents each token in slot s's textual description and seq^s_{max} is the maximum length of the slot token sequence. $T_s = [se^1, ..., se^{seq^s_{max}}]$ are token-level embedding of each slot token. Similar to E_s, E^{if}_s and E^{ds}_s represent the inform-slot and refer-slot contextualised embeddings. T^{if}_s and $T^{ds}_s \in \mathcal{R}^{seq^s_{max} \times n}$ are the token-level embedding of each slot token for inform-slot and refer-slot representations, respectively. By adding the special $[INFORM]$ and $[REFER]$ tokens in front of the slot's description, we obtain different representations

[1]Note that we can use any language models as the dialogue and slot encoders. For simplicity, we exploit BERT as the dialogue and slot encoders because it is widely used in previous literature [7, 23]

Figure 2: The architecture of Similarity-based Multi-Domain Dialogue State Tracking model (SM-DST).



of slot s, which can be used in different purposes. Indeed, the slot embeddings E_s and dialogue embeddings $E_{c,t}$ will be used by the span-based value prediction, which extracts the slot value from the whole dialogue context c_t. The inform-slot and refer-slot embeddings, E_s^{if} and E_s^{ds}, will be used by the slot operation predictor and refer-based value predictor, which extracts the slot value based on the auxiliary features.

3.4 The Dialogue and Slot Correlation Estimator

Unlike previous DST models [1, 7, 11] that directly pass the sentence-level representation e_t^{CLS} and the token-level representation $T_{c,t}$ to the predictors (the black lines in the left part of Figure 2), we introduce a dialogue and slot correlation estimator that aims to estimate the correlation scores between the current dialogue context c_t and the domain-slot pair s using their token-level contextualised representations, $T_{c,t}$ and T_s. The correlation estimator (green MaxSim rectangles in Figure 2) captures the relationship between dialogue context and candidate slots. For example, given a context "I'm looking for somewhere to stay in north Cambridge" and two candidate domain-slot pairs "taxi-destination" and "hotel-area", the estimator generates higher correlation scores to "hotel-area" than "taxi-destination" given the context. Unlike existing scalable DST models[13, 19] that learn the correlation between dialogue context and slots using their sentence-level representations (i.e. e_t^{CLS} and se^{CLS}), we instead learn the correlation using their token-level representation (i.e. T_s and $T_{c,t}$). Inspired by [10], given token-level representations of the dialogue context T_t^d and token-level representation of candidate slot s, T_s, the correlation scores of dialogue context d to slot s, $S_{c,s} \in \mathcal{R}^{seq^c_{max} \times seq^s_{max}}$, are estimated as $S_{c,s} = T_{c,t} \cdot T_s^T$, where the similarity between each dialogue's token-level contextualised embedding e_t and each slot's token-level embedding se is computed using cosine similarity. The similarity scores $S_{c,s}$ are then passed to the span-based value predictor described in Section 3.6, which extracts the slot values from the dialogue context c at turn t. We compute the inform- and refer-dialogue-slot correlation scores, $S_{c,s}^{if}$ and $S_{c,s}^{ds}$, of slot s to dialogue context c using the token-level inform- and refer-slot embeddings, T_s^{if} and T_s^{ds}, as:

$$S_{c,s}^{if} = \max_{e_t \in T_{c,t}} T_s^{if} \cdot e_t \cdot m_{s,t}^{if}, \tag{3}$$

$$S_{c,s}^{ds} = \max_{e_t \in T_{c,t}} T_s^{ds} \cdot e_t \cdot m_{s,t-1}^{ds}, \tag{4}$$

where $m_{s,t}^{if} \in \{0, 1\}$ is the system inform memory that indicates whether a value of slot s is informed by the system in turn t and $m_{s,t-1}^{ds} \in \{0, 1\}$ is the dialogue state memory that indicates whether a value of slot s is previously predicted by the DST model in the previous turn $t-1$. The correlation scores $S_{c,s}^{if}$ and $S_{c,s}^{ds} \in \mathcal{R}^{seq^s_{max}}$ will be used by the slot operation predictor (Section 3.5) and the refer-based value predictor (Section 3.7).

3.5 The Slot Operation Predictor

The slot operation predictor (the blue rectangle in Figure 2) predicts an operation for each slot to be one of: $O_{slot} = \{$ *none*, *dontcare*, *true*, *false*, *span*, *refer*, *inform*$\}$. The *none* and *dontcare* operations denote that the slot does not take a value or could be any value, respectively. The *true* and *false* operations denote a value of the given boolean slot, e.g., hotel-parking and hotel-internet.

The *span* operation denotes that a value of the given slot could be extracted from the current dialogue context c_t (see the value of hotel-area slot of turn 1 in Figure 1) and the model will obtain the slot value from the span-based value predictor (Section 3.6). The *refer* operation denotes that a value of the given slot could be referred from a slot value previously predicted in the previous turn m_{t-1}^{ds} (see turn 3 in Figure1 where the user refers the hotel name as the taxi departure). The *inform* operation denotes that a value of the given slot could be copied from the system inform memory m_t^{inform} (see turn 2 in Figure 1 where the user selects the hotel name from the system's recommendation). If the slot operation predictor says that a value of the given domain-slot pair can be referred or copied from the dialogue state memory m_{t-1}^{ds} or the system inform memory m_t^{inform}, then the model will obtain the value from the auxiliary-based predictors (Section 3.7).

The input to the slot operation predictor is the concatenation of the aggregate dialogue representation e_t^{CLS}, the inform-dialogue-slot correlation score $S_{c,s}^{if}$, and the refer-dialogue-slot correlation score $S_{c,s}^{ds}$. The probability distribution over the slot operations O_{slot} for domain-slot pair s at turn t is defined as follow:

$$\hat{y}_{t,s}^{slot} = softmax(W^{slot} \cdot (e_t^{CLS} \oplus S_{c,s}^{if} \oplus S_{c,s}^{ds})^\top + b^{slot}) \in \mathcal{R}^{|O_{slot}|} \tag{5}$$

where $W^{slot} \in \mathcal{R}^{|O_{slot}| \times d}$ and b^{slot} are learnable parameters and bias. The previous domain-specific DST models (e.g. [1, 7]) have independent weights W_s^{slot} and bias b_s^{slot} for each slot s. Our proposed SM-DST has only one weight W^{slot} and bias b^{slot} shared across slots. This allows SM-DST to transfer knowledge to new domains and unseen slots. The cross-entropy loss function for the slot operation prediction is defined as:

$$\mathcal{L}_{slot} = \sum_{t=1}^{T} \sum_{s=1}^{N} -log(y_{t,s}^{slot} \cdot (\hat{y}_{t,s}^{slot})^{\top}), \tag{6}$$

where $y_{t,s}^{slot}$ is the one-hot slot operation label for domain-slot pair s at turn t.

3.6 The Span-based Value Predictor

The span-based value predictor provides a slot value from the current dialogue context c_t by copying to give a value for each domain-slot pair from the dialogue context (the yellow rectangle in Figure 2). The span-based value predictor takes the token-level representation of the correlation between dialogue context c_t and slot s, $S_{c_t,s} \in \mathcal{R}^{seq_{max}^{c} \times n}$, as input and applies a two-way linear mapping to compute the probability of the terms being the start and the end position of the span for slot s, $\hat{y}_{t,s}^{start}$ and $\hat{y}_{t,s}^{end}$, as:

$$[\alpha, \beta] = W^{span} \cdot S_{c_t,s} + b^{span} \in \mathcal{R}^{seq_{max}^{c} \times 2},$$
$$\hat{y}^{start} = softmax(\alpha),$$
$$\hat{y}^{end} = softmax(\beta),$$

where $W^{span} \in \mathcal{R}^{seq_{max}^{s} \times 2}$ and $b^{span} \in \mathcal{R}^2$ are learnable parameters and bias. That the previous domain-specific DST models (e.g. [1, 7]) have independent weight W_s^{span} and bias b_s^{span} for each slot s. In contrast, the proposed SM-DST has only one weight W^{span} and bias b^{span} shared across different slots. This technique uses fewer parameters and allows SM-DST to transfer knowledge to new domains and unseen slots. Then, similar to the slot operation predictor's loss function (Equation (6)), we define the loss function for the span-based value prediction as:

$$\mathcal{L}_{span} = \sum_{t=1}^{T} \sum_{s=1}^{N} \frac{-log(y_{t,s}^{start} \cdot (\hat{y}_{t,s}^{start})^{\top}) - log(y_{t,s}^{end} \cdot (\hat{y}_{t,s}^{end})^{\top}}{2}, \tag{7}$$

where $y_{t,s}^{start}$ and $y_{t,s}^{end}$ are the one-hot start and end position label for domain-slot pair s at turn t.

3.7 The Refer-based Value Predictor

Inspired by Heck et al. [7], we describe the last component of the SM-DST model, the refer-based value predictor that leverages additional information including the system inform memory m_t^{inform} and the dialogue state memory from the previous turn m_{t-1}^{ds}, to effectively track the current dialogue state at turn t. Figure 1, the slot value of $hotel-name = LovellLodge$ can be referred to by using the system inform memory from Turn 2 and the value of $taxi-departure$ slot in Turn 3 should be inferred from the $hotel-name$ slot using the dialogue state memory from Turn 2. The input of the inform-based value predictor is the concatenation of the aggregated dialogue representation e_t^{CLS}, the refer-dialogue-slot similarity $S_{s,c}^{refr}$ and the inform-dialogue-slot similarity $S_{s,c}^{if}$. We determine the probability distribution over a particular domain-slot pair that its value that can be referred to is computed as:

$$\hat{y}_{t,s}^{refer} = softmax(W^{refer} \cdot (e_t^{CLS} \oplus S_{c,s}^{ds} \oplus S_{c,s}^{if})^{\top} + b^{refer}). \tag{8}$$

Similar to the weight and bias parameters in the slot operation and span-based value predictors, weight W^{refer} and bias b^{refer} parameters are shared across different domain-slot pairs. The loss function for the refer-based value predictor is computed as:

$$\mathcal{L}_{refer} = \sum_{t=1}^{T} \sum_{s=1}^{N} -log(y_{t,s}^{refer} \cdot (\hat{y}_{t,s}^{refer})^{\top}), \tag{9}$$

where $y_{t,s}^{refer}$ is the one-hot reference label for domain-slot pair s at turn t. We train all the components of model using the following joint loss function.

$$\mathcal{L}_{SM-DST} = \mu_{slot} \cdot \mathcal{L}_{slot} + \mu_{span} \cdot \mathcal{L}_{span} + \mu_{refer} \cdot \mathcal{L}_{refer}, \tag{10}$$

where μ_{slot}, μ_{span} and μ_{refer} are hyperparameters that control the weights of slot-, span- and refer-based loss functions.

4 EXPERIMENTAL SETUP

4.1 Datasets and Metrics

In particular, MultiWOZ2.1, MultiWOZ2.2 and MultiWOZ2.3 are the largest datasets and contain over 10,000 dialogues across seven domains: restaurant, taxi, attraction, hotel, train, hospital and police. MultiWOZ2.3 is the most recent benchmark dataset for DST[2]. It is an enriched version of MultiWOZ2.1 and MultiWOZ2.2 that identifies and fixes dialogue state annotation errors, inconsistencies and ontology related issues from MultiWOZ2.1. Following [11, 20, 23], we remove hospital and police domains in MultiWOZ2.1, MultiWOZ2.2 and MultiWOZ2.3 because they only appear in the training dataset. We use the standard training/validation/test split strategy provided in the original datasets. Following previous literature [1, 7, 11, 23], we use two evaluation metrics to evaluate the effectiveness, joint goal accuracy and slot accuracy [8]. At each turn, the joint goal accuracy is 1.0 if and only if all domain-slot pairs and their corresponding values are correctly predicted, otherwise 0. The score is averaged across all turns in the test set. The slot accuracy individually compares each domain-slot pair and its corresponding value to the ground truth label. We conduct statistical significance testing using a paired t-test and report significance at the 95% confidence level.

4.2 Baseline Models

We implement our proposed SM-DST model using PyTorch[3]. We use the pre-trained BERT-base-uncased model[4] that has 12 hidden layers with embedding dimension $n = 768$ as the dialogue encoder[4]. For all previous DST baseline systems, we optimise them similarly using cross-entropy loss and the Adam optimiser [12] with a learning rate of $2e^{-5}$. For the hyperparameters, we use the optimized parameters reported in their previous work. A summary of the baselines is described below.

[2]https://github.com/lexmen318/MultiWOZ-coref
[3]https://github.com/feay1234/SM-DST
[4]https://huggingface.co/transformers/pretrained_models.html

BERT-DST[1] exploits a pre-trained BERT model to encode the utterance and system response into sentence- and token-level representations that are used for the slot operation and slot value predictions.

TRADE[19] is the scalable DST model that encodes the whole dialogue context using bidirectional Gated Recurrent Units (GRU) [3] and generates the value for every slot using the GRU-based copy mechanism.

MA-DST[13] is the state-of-the-art scalable DST model that exploits deep contextualised word representations (ELMO) [17] to learn relationships between dialogue context and slots.

SUMBT[15] is an ontology-based DST model that exploits BERT as the encoder for the dialogue context and domain-slot pairs. After encoding them it scores every candidate slot value in a non-parametric manner using a distance measure.

Picklist-DST[23] is the state-of-the-art ontology-based DST model that requires a pre-defined ontology with all possible values for each domain-slot pair.

DS-DST [23] is a hybrid model that jointly trains both the ontology- and span-based model. The ontology-based model used in DS-DST is Picklist-DST and the span-based model is similar to BERT-DST.

SOM-DST [11] is a span-based model that consists of the slot operation predictor and the slot value generation[5]. It uses the copy-mechanism for the slot operation prediction and uses Gated Recurrent Units (GRU) [3] for the slot value prediction.

SST [2] is a schema-guided model that exploits graph attention networks to learn the slot-slot relationships from the utterances and pre-defined schema graph.

TripPy [7] is the state-of-the-art span-based DST model that uses the triple copy mechanisms to track the dialogue state.[6]

5 EXPERIMENTAL RESULTS

5.1 Baseline Comparison

Tables 1 shows the effectiveness of DST models in terms of joint goal accuracy on the three MultiWOZ datasets. The encoder column indicates the pre-trained language model used by the baselines as the dialogue encoder. Due to their recency or a lack of details, we were not able to re-implement all baselines. For comparison we include the as-reported results and are unable to test statistical significance. We reproduce the results for TripPy and SOM-DST models in Table 1. We find that the relative dialogue state tracking effectiveness of these two models on MultiWOZ2.1 is consistent with the results reported in the original papers [7, 11]. SOM-DST outperforms both TRADE and DS-DST and is as effective as the state-of-the-art ontology-based DST model (DST-picklist). Among all the baselines in Tables 1, we observe that TripPy outperforms all the ontology-based and span-based baselines on MultiWOZ2.1. MultiWOZ2.2 and MultiWOZ2.3 are the most recent DST datasets and are not been widely compared to in the previous literature. As a result, comparison is only available on a subset. The results of TRADE, SUMBT and DS-DST on MultiWOZ2.2 and MultiWOZ2.3 are those reported in [6, 22]. We do not report results of SOM-DST on MultiWOZ2.2 as its implementation does not support it.

[5]https://github.com/clovaai/som-dst

[6]https://gitlab.cs.uni-duesseldorf.de/general/dsml/trippy-public

Table 1: Joint Goal Accuracy of DST models on the MultiWOZ2.1, MultiWOZ2.2 and MultiWOZ2.3 datasets. The best result is highlighted in bold and $*$ denotes a significant difference between the best and the second-best performing results according to a paired t-test at $p < 0.05$. † indicates a result previously reported.

Model	Encoder	MultiWOZ2.1	MultiWOZ2.2	MultiWOZ2.3
TRADE†	GRU	45.60	45.40	49.20
SUMBT†	GRU	49.20	49.70	52.90
DS-DST†	BERT	51.21	51.70	-
MA-DST†	ELMO	51.04	-	-
SOM-DST	BERT	52.37	-	55.50
DST-picklist†	BERT	53.30	-	-
SST†	GAMT	55.23	-	-
TripPy	BERT	55.52	50.71	58.80
SM-DST	BERT	**56.86***	**53.82***	**62.44***

Comparing our proposed SM-DST model with the baselines across the three multi-domain DST datasets in Table 1, we see that it consistently outperforms all the ontology- and span-based DST baselines in terms of joint goal accuracy across all datasets. SM-DST significantly outperforms the state-of-the-art DST model, TripPy, by 2.4%, 6.1% and 6.2% relative reduction in error on MultiWOZ2.1, MultiWOZ2.2, and MultiWOZ2.3. We believe SM-DST is more effective than TripPy for several reasons. First, SM-DST shares tracking knowledge across domains, while TripPy only learns per-domain. Second, SM-DST learns the domain-slot similarity leveraging their token-level representations (described in Section 3.4) while TripPy uses coarser sentence-level representations that we hypothesize does not fully capture the nuance of domain-slot relationships. Comparing SM-DST with the state-of-the-art scalable DST model, MA-DST, it outperforms MA-DST by 13.6% relative reduction in error on MultiWOZ2.1. Although the architecture of MA-DST shares the tracking knowledge across domains, it does not leverage the system inform memory $m^{if}_{s,t}$ or the dialogue state memory $m^{ds}_{s,t-1}$, which we believe hurts its effectiveness. In contrast, SM-DST exploits the refer-based value predictor, described in Section 3.7, that leverages the system inform memory $m^{if}_{s,t}$ and the dialogue state memory $m^{ds}_{s,t-1}$ to track the current dialogue state. For a detailed breakdown of the contribution of the components we refer the reader to the detailed ablation study on each component of SM-DST conducted in previous work [7].

5.2 Few-shot DST Effectiveness

Collecting training data for new domains is expensive and time-consuming [9, 14, 18]. In this section, we compare the effectiveness of SM-DST and TripPy in the few-shot experimental setup. We study whether SM-DST is effective in tracking the dialogue state even when the training dataset for the new domain is very limited. We assume that just 1% of the original training data for the new domain is available (around 20 dialogues). We note that TripPy is capable of the few-shot setting, although the parameters of its predictors are separate for each slot, its dialogue encoder is shared across different domains (see Section 3.2). Therefore, if we have a few training dataset for TripPy to train their predictors, it should be able to transfer the tracking knowledge to unseen domains. Table 2 shows

Table 2: As per Table 1. Joint Goal Accuracy of TripPy and SM-DST models on the MultiWOZ2.1, MultiWOZ2.2 and MultiWOZ2.3 datasets in the few-shot setting. $*$ denotes a significant difference with a paired t-test at $p < 0.05$.

	MultiWOZ2.1		MultiWOZ2.2		MultiWOZ2.3	
Domain	TripPy	SM-DST	TripPy	SM-DST	TripPy	SM-DST
Hotel	69.34	**74.17***	69.59	**71.70***	69.03	**74.70***
Train	76.55	**81.96***	67.63	**86.46***	77.55	**85.51***
Taxi	91.32	**91.52**	91.29	**91.59**	91.66	**91.78**
Restaurant	76.22	**86.88***	78.89	**87.70***	80.91	**88.17***
Attraction	77.55	**90.37***	79.04	**89.47***	81.02	**88.31***

the joint goal accuracy of SM-DST and TripPy across five domains on the MultiWOZ datasets for the few-shot setting. We observe that SM-DST consistently outperforms TripPy on all domains across the three datasets. SM-DST significantly outperforms TripPy on the hotel, train, restaurant and attraction domains by 5-12% on MultiWOZ2.1, 2-18% on MultiWOZ2.2, and 5-8% on MultiWOZ2.3. Interestingly, on the restaurant and attraction domains using only around 20 training dialogues for these two domains to train the model obtain almost 90% of the full dataset, while TripPy can only achieve around 80%. This indicates that the model requires less data than TripPy to reach the same level of effectiveness. We also observe that the behavior of TripPy and SM-DST on the taxi domains are similar. We believe the reason is because all four taxi slots (i.e. leaveAt, departure, destination and arriveBy) share similar values with the corresponding slots in the train domain.

Next, we investigate the effectiveness of SM-DST and TripPy in detail for each slot in the few-shot setting. The results in Table 3 show that SM-DST is consistently more effective than TripPy across all slots in the most recent and reliable dataset, MultiWOZ 2.3. There is approximately a 3% absolute improvement of SM-DST over TripPy on average across all domain-slot pairs across all versions of the dataset. TripPy only marginally outperforms SM-DST in three slots for MultiWOZ2.1 and 2.2. The reasons for this on the older datasets is unclear, but these had minor inconsistencies and annotation issues.

We see that SM-DST outperforms TripPy by approximately 5-7.5% absolute on the hotel-name, restaurant-name and attraction-name domain-slot pairs. The values of these domain-slot pairs are open vocabulary and are therefore more difficult to extract from the dialogue context [7, 13, 19]. The largest gains over TripPy are in hotel-internet and hotel-parking, with over an 8% improvement. The hotel-type slot remains the most challenging for both models, and although SM-DST improves by over 4.5% absolute, it is the worst performing slot. In cases where the slot semantics are similar (area, day) we also observe gains of SM-DST. On the unseen domain-slot pairs (e.g. hotel-type, hotel-parking, hotel-stars and hotel-internet)[7], we observe that both SM-DST also provides significant gains. The hotel-internet is 8.4%, parking is 8.2%, type is 4.6%, and stars is 3%. We hypothesize the reason for this may be due to the more granular per-token similarity used in SM-DST.

Furthermore, we analyse the DST predictions of TripPy and SM-DST in the few-shot setting on MultiWOZ2.3 in Table 4. In the first row of Table 4, we observe that both TripPy and SM-DST can effectively track the user's dialogue state for the similar slots (e.g. the departure, destination and leaveAt slots that are shared between the train and taxi domains). However, in the second row, we find that TripPy makes a mistake in extracting the value of the taxi-departure slot, while SM-DST does not. These results show that SM-DST is more effective and consistent than TripPy in tracking the user's dialogue state for the similar slots. Both TripPy and SM-DST share similar mistakes when dealing with the unseen slots (e.g. the restaurant-area slot which is completely new to both models). Since we only train these two models on the few-shot dataset (i.e. 1% of the original dataset for the restaurant domain), they may not have seen the restaurant-area slot during the training process. In the last row of Table 4, we observe another mistake of TripPy when it extracts the value of the attraction-type slot, whereas SM-DST accurately predicts the value of the attraction-type slot. Overall, in response to research question RQ2, the results reported in this section demonstrate that SM-DST transfers more effectively on slots that may be similar across domains as well as ones that have more limited training data.

Table 3: Effectiveness of TripPy and SM-DST models in terms of slot accuracy on the MultiWOZ datasets in the few-shot setting.

	MultiWOZ2.1		MultiWOZ2.2		MultiWOZ2.3	
domain-slot pair	TripPy	SM-DST	TripPy	SM-DST	TripPy	SM-DST
attraction-area	94.13	**96.47**	93.95	**96.55**	94.75	**96.69**
attraction-name	85.54	**96.69**	86.94	**96.70**	89.64	**97.08**
attraction-type	90.38	**95.77**	91.89	**94.89**	91.63	**92.95**
hotel-area	94.48	**95.89**	93.79	**95.89**	93.91	**96.16**
hotel-day	98.09	**98.86**	98.44	**98.77**	98.25	**98.72**
hotel-people	97.50	**97.94**	98.06	**98.22**	97.83	**97.88**
hotel-stay	98.44	**98.98**	98.32	**99.31**	98.75	**99.06**
hotel-internet	85.17	**94.72**	**84.63**	**84.63**	88.36	**96.73**
hotel-name	89.95	**96.54**	90.97	**97.41**	92.09	**97.57**
hotel-parking	85.40	**91.24**	85.40	**85.40**	84.37	**92.53**
hotel-price	95.27	**97.22**	95.16	**96.85**	95.67	**97.27**
hotel-stars	94.53	**97.71**	95.48	**97.98**	94.98	**98.03**
hotel-type	82.00	**85.57**	82.73	**85.89**	78.46	**83.04**
restaurant-area	94.57	**96.85**	94.15	**96.70**	95.28	**96.81**
restaurant-day	98.39	**98.41**	**98.68**	98.43	98.25	**98.59**
restaurant-people	**98.32**	98.25	**98.30**	98.06	98.81	**98.86**
restaurant-time	**98.77**	98.68	98.47	**98.78**	98.62	**99.00**
restaurant-food	94.61	**97.12**	94.45	**97.65**	96.46	**97.68**
restaurant-name	86.88	**95.71**	89.53	**97.10**	91.35	**96.45**
restaurant-price	94.11	**97.82**	94.32	**97.34**	94.61	**98.02**
taxi-arriveBy	96.79	**96.80**	96.80	**97.46**	96.62	**96.91**
taxi-departure	92.50	**94.18**	92.47	**93.08**	93.64	**93.65**
taxi-destination	92.17	**93.00**	92.08	**92.93**	93.23	**93.68**
taxi-leaveAt	96.80	**97.44**	96.31	**98.24**	97.64	**98.58**
train-arriveBy	93.53	**95.48**	87.32	**98.26**	89.64	**95.65**
train-people	95.52	**96.36**	94.99	**96.19**	96.32	**96.53**
train-day	96.46	**98.40**	95.58	**97.63**	97.64	**98.63**
train-departure	90.93	**97.23**	92.72	**96.64**	94.91	**97.21**
train-destination	95.98	**98.05**	94.68	**97.73**	96.55	**98.17**
train-leaveAt	**92.06**	90.37	81.78	**96.64**	90.98	**94.00**
average	93.31	**96.13**	92.95	**95.91**	93.97	**96.40**

[7] These slots do not overlap with other slots from different domains

Table 4: Example dialogue states predicted by TripPy and SM-DST on the few-shot setting on MultiWOZ2.3. The accurate predicted values are highlighted in green, otherwise in red.

Dialogue Context	TripPy	SM-DST
Dialogue id : SNG0448 **Usr**: I'd like a train from Cambridge to Broxbourne, please. **Sys**: There are 133 trains making that trip, do you have a day and time you 'd like to go? **Usr**: Yes . I would like to leave on Sunday after 20:30. **Sys**: TR7208 leaves at 21:01 . Would you like me to book it? **Usr**: Yes and could I have the reference number after?	train-departure : Cambridge train-destination : Broxbourne train-day : Sunday train-leaveAt : 21:01	train-departure : Cambridge train-destination : Broxbourne train-day : Sunday train-leaveAt : 21:01
Dialogue id : SNG02207 **Usr**: I need to book a tax departing from gandhi.	taxi-departure : none	taxi-departure : gandhi
Dialogue id : MUL0034 **Usr**: I'm looking for an expensive restaurant that serves Thai food, please **Sys**: There is one in the west and one in the centre of town . Do you have a preference in area ? **Usr**: I don't have a preference in the area. I suppose you could book me a table for two people at 12:30 on Saturday.	restaurant-area : none restaurant-people : two restaurant-data : Saturday restaurant-price : expensive restaurant-time : 12:30	restaurant-area : none restaurant-people : two restaurant-data : Saturday restaurant-price : expensive restaurant-time : 12:30
Dialogue id : PMUL4106 **Usr**: Hello, I am looking for a concerthall in the centre of town.	attraction-type : hall attraction-area : centre	attraction-type : concerthall attraction-area : centre

5.3 Zero-shot DST Effectiveness

In this section, we study the ability of SM-DST to generalise to unseen domains by considering the zero-shot setting. Zero-shot assumes that there is no training data for a new domain. For example, on the MultiWoZ2.1 dataset, we train the models on the hotel, train taxi and restaurant domains and evaluate their performance on the attraction domain. This setting is more challenging than the few-shot setting because models can only rely on parameters learned from the seen domains and can not learn per-domain parameters. Note that TripPy does not support the zero-shot setting, we therefore do not report results for it in the setup. We show the results comparing SM-DST with state-of-the-art zero-shot models, TRADE and MA-DST in Table 5. We only report effectiveness on MultiWOZ2.1 for compatibility with both models. As expected, the effectiveness of all models drops dramatically. Overall, we observe that SM-DST consistently outperforms these methods on MultiWOZ2.1. The largest and most significant gain over existing models is in the train domain, with a 25% absolute effectiveness gain. Other domains show consistent improvement between 3-5%. Similar to previous results in [13, 19], we see that SM-DST performs the best on the taxi domain. We hypothesize that reason why the zero-shot performance of SM-DST on the taxi domain is relatively high is because all the four slots (taxi-arriveBy, taxi-leaveAt, taxi-departure and taxi-destination) share similar values with the corresponding slots in the train domain (train-leaveAt, train-arriveBy, train-departure, train-destination). This also explains why our model makes such large improvements in the train domain, because it can leverage the similarity to taxi slots. Overall, in response to research question RQ3 the results demonstrate that SM-DST can effectively transfer the tracking context across slots and domains with similar semantics.

Table 5: As per Table 1, JGA of scalable DST models in the zero-shot setting on the MultiWOZ2.1 dataset.

Domain	TRADE†	MA-DST†	TripPy	SM-DST
Hotel	14.20	16.28	-	**23.00**
Train	22.39	22.76	-	**47.92**
Taxi	59.21	59.27	-	**62.90**
Restaurant	12.59	13.56	-	**16.82**
Attraction	20.06	22.46	-	**27.29**

6 CONCLUSION

We address the problem of scaling and domain transfer for multi-domain task-based virtual assistants. We propose a novel Similarity-based Multi-Domain Dialogue State Tracking model (SM-DST). The architecture of the model consists of five key components that have shared parameters across slots and domains. This allows it to leverage data and parameters across domains and slots that share similar semantics (e.g. train and taxis). The comprehensive experiments on the DST benchmark datasets show that SM-DST significantly outperforms the current state-of-the-art DST model, TripPy, by approximately 2-6% relative reduction in error in the standard supervised setting. In the zero-shot setting, SM-DST outperforms the state-of-the-art model, MA-DST, by 3-25% absolute on joint goal accuracy for MultiWOZ2.1.

ACKNOWLEDGEMENTS

This project was funded by the EPSRC Fellowship titled "Task Based Information Retrieval", grant reference number EP/P024289/1, and supported by the Engineering and Physical Sciences Research Council grant EP/V025708/1 as well as the Bloomberg Data Science Research Grant.

REFERENCES

[1] Guan-Lin Chao and Ian Lane. 2019. BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer. In *INTERSPEECH*.

[2] Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-Guided Multi-Domain Dialogue State Tracking with Graph Attention Neural Networks.. In *Proc. of AAAI*.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

[5] M. Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, A. Sethi, Anuj Kumar Goyal, Peter Ku, Sanchit Agarwal, Shuyang Gao, and Dilek Z. Hakkani-Tür. 2019. MultiWOZ 2.1: Multi-Domain Dialogue State Corrections and State Tracking Baselines. *ArXiv* abs/1907.01669.

[6] Ting Han, Ximing Liu, Ryuichi Takanobu, Yixin Lian, Chongxuan Huang, Wei Peng, and Minlie Huang. 2020. MultiWOZ-coref: A multi-domain task-oriented dataset enhanced with annotation corrections and co-reference annotation. *arXiv preprint arXiv:2010.05594*.

[7] Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking. In *Proc. of SIGdial*.

[8] Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proc. of SIGDIAL*. 263–272.

[9] Yiping Kang, Yunqi Zhang, Jonathan K Kummerfeld, Lingjia Tang, and Jason Mars. 2018. Data collection for dialogue system: A startup perspective. In *Proc. of NAACL*. 33–40.

[10] Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proc. of SIGIR*. 39–48.

[11] Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-woo Lee. 2020. Efficient Dialogue State Tracking by Selectively Overwriting Memory. In *Proc. of ACL*.

[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

[13] Adarsh Kumar, Peter Ku, Anuj Goyal, Angeliki Metallinou, and Dilek Hakkani-Tur. 2020. MA-DST: Multi-Attention-Based Scalable Dialog State Tracking. In *Proc. of AAAI*, Vol. 34. 8107–8114.

[14] Walter Lasecki, Ece Kamar, and Dan Bohus. 2013. Conversations in the crowd: Collecting data for task-oriented dialog learning. In *Proc. of AAAI*, Vol. 1.

[15] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-Utterance Matching for Universal and Scalable Belief Tracking. In *Proc. of ACL*. 5478–5483.

[16] Jarana Manotumruksa, Jeff Dalton, Edgar Meij, and Emine Yilmaz. 2021. Improving Dialogue State Tracking with Turn-based Loss Function and Sequential Data Augmentation. In *Findings of EMNLP*. 1674–1683.

[17] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[18] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

[19] 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 *Proc. of ACL*.

[20] 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 *Proc. of ACL*.

[21] Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. 2021. Slot Self-Attentive Dialogue State Tracking. In *Proc. of WWW*. 1598–1608.

[22] 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 *Proc. of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL*. 109–117.

[23] Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*.

[24] Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. In *Proc. of ACL*.