TASK ORIENTED DIALOG SYSTEMS

A Thesis

by

SANUJ SHARMA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Ruihong Huang
Committee Members,	Anxiao Jiang
	Vinayak Krishnamurthy
Head of Department,	Scott Schaefer

May 2020

Major Subject: Computer Science

ABSTRACT

Task oriented dialog systems hold numerous applications in assisting users to achieve various goals. They often comprise of a pipeline of individual components. In this work, our contribution is towards two such components, namely, dialog state tracker and natural language generator. A typical conversation comprises of multiple turns between participants where they go back-and-forth between different topics. At each user turn, dialogue state tracking (DST) aims to estimate user's goal by processing the current utterance. However, in many turns, users implicitly refer to the previous goal, entailing the use of relevant dialogue history. Nonetheless, distinguishing relevant history is challenging and a popular method of using dialogue recency for that is inefficient. We, therefore, propose a novel framework for DST that identifies relevant historical context by referring to the past utterances where a particular slot-value changes and uses that together with weighted system utterance to identify the relevant context. Specifically, we use the current user utterance and the most recent system utterance to determine the relevance of a system utterance. Furthermore, we do empirical analyses to show that our method improves joint goal accuracy on WoZ 2.0 and MultiWoZ 2.0 restaurant domain datasets respectively over the previous state-of-the-art models. Secondly, we study a family of deep generative models for generating system response in a task oriented dialog setting. The language generation tasks involves conditioning the output of the generative models on the current dialog state, system act and the previous user utterance. Finally, we do qualitative analysis and report the perplexity scores for a transformer encoder-decoder model and a conditional variational auto-encoder on schema guided dialog state tracking dataset.

ACKNOWLEDGMENTS

I would like to thank all the people involved in helping me with my research and supporting me while at Texas A&M University. Firstly, I am grateful to my Committee Advisor Dr. Ruihong Huang for providing me the opportunity to work under her guidance on challenging problems and guiding me throughout the research process. I would like to thank my other Committee members including Dr. Anxiao Jiang and Dr. Vinayak Krishnamurthy for their valuable comments and constructive feedback on my thesis. Secondly, I'm thankful to my colleagues in the natural language processing lab for their moral support, the reading group discussions and friendship. I'm specifically grateful to Prafulla Choubey for helping me get started in NLP and pointing me towards the important problems in the field. Finally, I would like to thank my family and friends for their support and encouragement.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Ruihong Huang [advisor] and Professor Anxiao Jiang of the Department of Computer Science and Professor Vinayak Krishnamurthy of the Department of Mechanical Engineering.

Prafulla Choubey, who is a Ph.D. student at the Department of Computer Science, was a contributor on research conducted on Dialog State Tracking. All other work conducted for the thesis was completed by the student independently.

Funding Sources

Travel to NAACL 2019 (2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics) was supported by a travel grant from the department of Computer Science and Engineering at Texas A&M University. There is no outside funding sources to acknowledge for the research and completion of this thesis.

NOMENCLATURE

DST	Dialog State Tracking
LSTM	Long Short Term Memory
GLE	Global-local self-attentive encoder
CVAE	Conditional Variational Autoencoder
WoZ	Wizard-of-Oz System
ASR	Automatic Speech Recognition
NLU	Natural Language Understanding
NLG	Natural Language Generation
TTS	Text to Speech

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION

Natural language and conversation are important characteristics of human intelligence which allows us to communicate complex ideas on a daily basis. Dialog Systems or conversational agents are computer programs which aspire to communicate with the user (or a group of users) in natural language conversational format. They are broadly categorized into two classes, namely, chatbots and task-oriented dialog agents. **Chatbots** aim to mimic unstructured human conversations by conducting casual chit-chat discussions about different topics. These dialog agents don't have a practical utility and are often designed for entertainment purposes. On the other hand, you have **task-oriented dialog agents** which help the users to complete a set of tasks by carrying out conversations. Some real world examples of task-oriented dialog agents are digital assistants like Google Assistant, Alexa, Siri and Cortona which assists the user by scheduling events in the calendar, booking cabs and restaurants, giving directions etc.

Even with the recent progress in natural language processing (NLP), making computers talk like humans remains infeasible. A dialog can be defined as a sequence of **turns** from the participants, similar to how people take turns rolling a dice while playing a board game. A dialog agent is required to identify if the current turn is the last turn or not, in order to stop talking, also called endpoint detection. (8; 9) had an insight that each utterance in a dialog is kind of an action being performed by the speaker. These speech acts or **dialog acts** can be of different kinds but they fall into a small category of acts like answering, asking, planning, thanking etc. A dialog is not a sequence of independent speech acts but it maintains a structure with sub-dialogues. For instance questions are followed by answers or proposals are followed by rejections / acceptance. Moreover, there may be clarification questions and the speakers also establish a common ground on what they agree upon, by saying "Okay, ..." or repeating what the other person said, during a conversation. A discussion can also be controlled by the dialog agent, the user or the initiative may shift back and forth. The user often implicitly states some facts in an indirect manner or just assumes some facts to be common knowledge and this makes it difficult for the system to respond in a sensible manner.

These properties of human conversation adds complexities in designing any kind of dialog agent. For task oriented dialog agent, there is additional challenge of retrieving information from external knowledge bases.

There are three important architectures for designing chatbots: encoder-decoder models, information retrieval systems, and rule-based agents. Encoder-decoder models (10), inspired from phrase-based machine translation, train on a corpus and treat the implementation of a dialog agent like a transducing task from a particular context (last user utterance or dialog history encoding) to the system response. In chapter 5, we show how encoder-decoder models can also be utilized for dialog generation in a task-oriented setting. Information retrieval systems are also a corpus-based approach which look up the most similar turn from a knowledge base and use that to generate the system response. ELIZA (11) is considered one of the most important dialog systems in the field which is rule-based agent as opposed to corpus-based. It was designed to simulate a Rogerian psychologist which tries to draw the patient by reflecting their statements back at them. This allowed the dialog agent to know nothing about the real world. PARRY (12) was another rule-based dialog agent, like ELIZA, designed to study schizophrenia. It had rules to simulate different levels of fear and anger, and is the first known system to pass the Turing test (13).

Architectures for task-oriented dialog agents have evolved over time to employ a pipeline of multiple individual components designed for a specific purpose. GUS (Genial Understander System)) architecture (14) was one of the first frame-based architectures proposed for task-oriented dialog agents. It was designed for a travel planning task and still underlies some of the commercial digital assistants today. GUS uses **frames** to extract user's intentions from the utterances. A frame is a knowledge representation which consists of **slots** taking some pre-defined values supported by a **domain ontology**. The dialog agent's goal is to fill the slots with user's intended filters and perform the next action conditioned on the slot values. Usually, these actions are triggered based on hand-crafted rules based on the combination of slot-value pairs. These agents can also be multi-domain which requires a **domain classification** step before any of the slot-filling happens. For dialog generation, **templates** are used which are created by the dialog designer. Rule-based

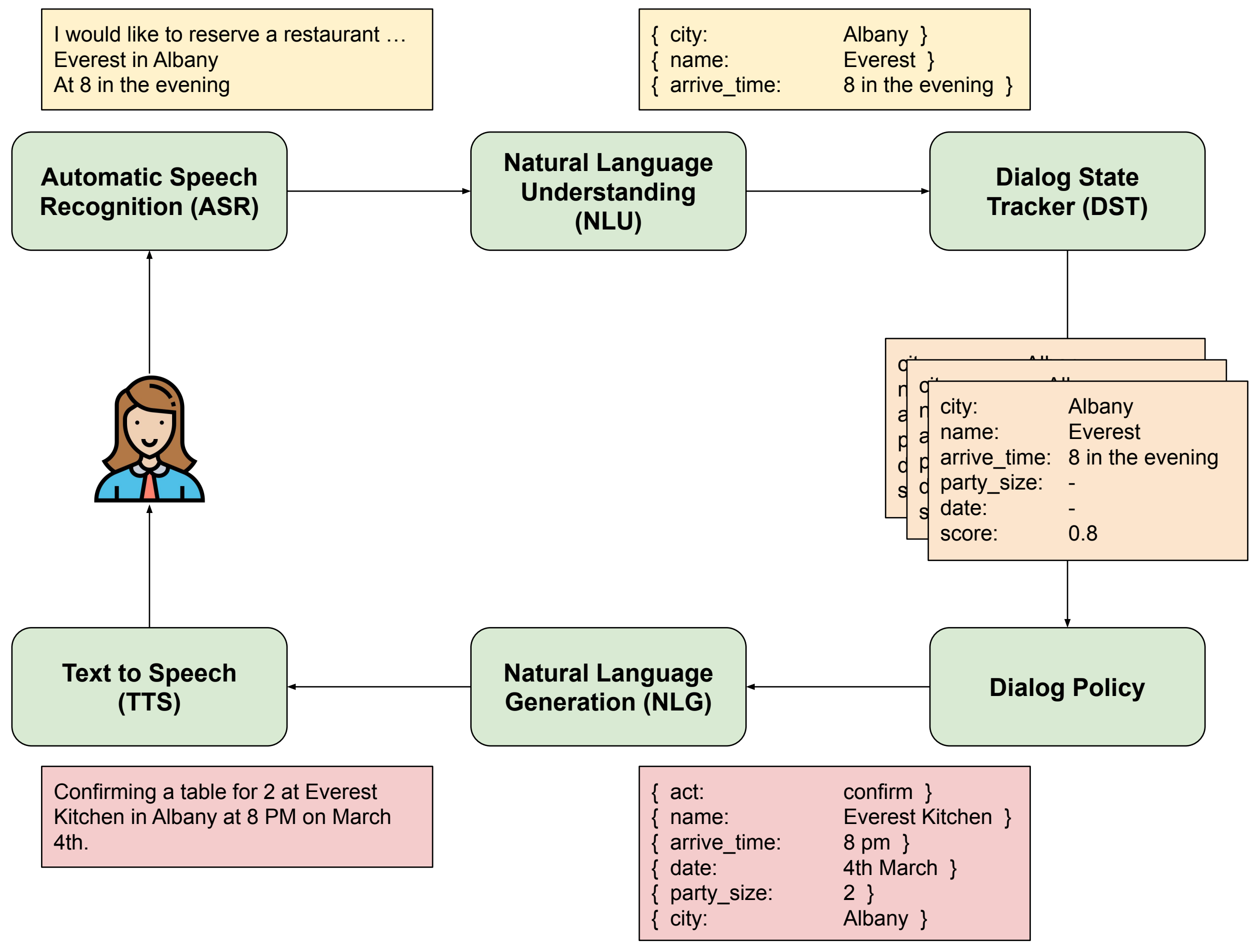

Figure 1.1: Dialog State Architecture. (1) Modern dialog systems consist of a pipeline of components (green boxes) responsible for tasks like ASR, NLE, DST, NLG and TTS. Yellow boxes show the processing of user's utterance to slot-filling for that turn. DST component keeps track of the cumulative dialog state (orange box). The dialog policy decides the next system act, the NLG component generates the utterance, which is finally converted to speech by a TTS component (red boxes show the process).

GUS architecture is heavily used in industry applications. This architecture requires experts to design the dialog agent which is restricted to a narrow domain. The hand-crafted rules are slow and expensive to create but they lead to high precision with a recall problem.

1.1 Dialog State Architecture

The dialog state architecture is a more sophisticated version of the frame-based architecture discussed in the previous section, and most modern task-oriented dialog agents are base on this architecture. The architecture is visualized in Figure 1.1. It consists of six components, namely, automatic speech recognition (ASR), natural language understanding (NLU), dialog state tracker (DST), dialog policy, natural language generation (NLG), and text to speech (TTS).

The user utterance is processed by an ASR component to convert the speech to textual format. The textual representation is fed into an NLU unit. The NLU unit is responsible for understanding the user's utterance and extracting the information into a more structured format. It may involve named-entity recognition, sentiment analysis, coreference resolution, domain classification etc. The DST takes the output from the NLU unit and uses that to update the cumulative dialog state until that turn. It consists of a slot-filling task to keep track of user's goals and intents, which also represents the current state of the conversation. The dialog policy component takes the dialog state as input and generates the next action, also called system act, for the dialog agent. The dialog policy decides whether the system should make a suggestion, ask for clarification, answer a question, change the topic and so on. It also interacts with external knowledge bases or production services to query more information to generate the system act. The system acts are given to the NLG component which converts the structured representation of system acts into a textual utterance which the dialog agent would use to respond to the user. Finally, the textual system response is converted to a speech utterance with a TTS component, which processes the text input to generate an utterance spoken like a human.

These six components are not entirely independent of each other in practice. For instance, the ASR accuracy is often improved by a language model which predicts the probability of next word given the previous ones. The language model can be used for NLU and DST as well. The dialog state and the previous system act can also be used to help the NLU component to understand the current utterance. For instance, if the system asked about the food preference of the user, then it's highly likely that the utterance is going to contain an answer for that, and the NLU component can be conditioned with the domain ontology to better deal with this scenario. Moreover, these components can be further dependent on other production services and the processing path from user utterance to system response generation is not always straightforward as shown in Figure 1.1.

Nevertheless, for the purpose of this dissertation, the implementation details are abstracted away and the datasets used for empirical analysis are loosely based on this architecture.

1.1.1 Dialog Acts

In dialog state architecture, a dialog act combines the concept of speech act and grounding. It represents the broad action of the current turn or utterance. A dialog act is often associated with the system utterance and hence it is also called system act. System act is the output of the dialog policy component. Figure 1.2 shows examples of system act for each system utterance or turn. There can be multiple system acts like *REQUEST*, *CONFIRM*, *INFORM*, *OFFER* and *GOODBYE*. Some acts also contain slot type-value pairs which are used to construct the system utterance. For instance, a system act like *OFFER(movie_name: Red Joan)* may generate a system utterance like *"Would you like to see the movie Red Joan?"*. The system acts are designed by the dialog designer and there is no known way to learn them directly from the corpus. They also depend on the domain, some acts may be common to multiple domains and some may be unique to particular domains, as defined in the ontology.

1.1.2 Dialog State Tracking

Dialog state (also called belief state) aims to keep track of cumulative user's goals and requests during the conversation. It consists of key-value pairs called slot type and slot value. Figure 1.2 shows a conversation where the user wants to do do restaurant reservation and enquire about a movie. For each user turn, it shows the dialog state with the blue boxes on the right. It consists of determining the active service (domain) and intent of the user. The user's utterance is used to fill slots that the system requires to proceed further with the conversation. When the user says "Could you help me with a restaurant reservation?", the system recognizes the dialog state as "service: restaurant, intent: reserve restaurant". In subsequent turns when the user says "Thank you for that. I'd like to see a movie too. ...", the dialog state shows the active service as movie and intent as "find movies". The system also fills the slot types like "genre: drama, location: Albany". A frame is created for each user turn and the slot types and values are extracted from the user utterance.

The current frame is then combined with the current dialog state by updating the slot types which have a different value in the current frame compared to the expired value of that slot type in the current dialog state.

The dialog state corresponding to a particular turn is expected to capture the state of the entire conversation including the current turn. It can be thought of as a sparse representation of the conversation which is used by the dialog policy to decide the next action for the dialog agent. The dialog state is also useful for the system to query an external knowledge base to get the required information to answer user's questions. It can also be used to call a production service to complete some action for the user like adding a calendar event, purchasing movie tickets, or booking a restaurant.

Similar to dialog acts, the slot types and slot values in the dialog state are also manually devised by the dialog designer instead of extracting them through corpus directly. Hence, the slot types and values are also dependent on the domain ontology.

1.1.3 Dialog Policy

The dialog policy generates the next system action given the previous system actions and the user utterances. More formally, it can be stated in the below equation where *A* is the system act, *U* is the user utterance, and their subscript represents the turn number.

$$A_i = \arg\max_{A_i \in A} P(A_i | A_1, U_1, ..., A_{i-1}, U_{i-1}) \tag{1.1}$$

Since we have the dialog state tracker, the above equation can be simplified to use the dialog state instead of the previous user utterances. Many techniques prefer to condition the policy model on the previous frames from the dialog state, the last system and user utterances. There are both policy learned from the corpus and rule-based policies but rule based policies are more common for industrial applications. More sophisticated models use reinforcement learning to train the policy model where the reward is given after each turn or at the end of the conversation (15). It remains a challenge to learn a good dialog policy from a corpus because of complex characteristics of human conversation, and it is also expensive to generate labeled data or devise dialog simulators for training the policy model.

1.1.4 Response Generation

The system response generation is done by the NLG component. Response generation requires to decide what to say (content planning) and how to say it (sentence realization). In the dialog state architecture, most of the content planning is carried out by the dialog policy. Therefore, the task of response generation boils down to generating natural language response given some context, which mostly includes the state of dialog state, system act, user utterance and so on. Certain models also utilize the dialog history to generate the response by maintaining a hidden state representation with a RNN or using the attention mechanism to summarize the representations for each historical turn.

As is the case with dialog state tracking, collecting data for response generation task is expensive and complicated. The dataset should include labeled dialog state, system acts, the dialog history along with the utterances for each turn. System acts should also include relevant slot type and values. Different datasets follow different conventions and there is added variance for multi-domain datasets with large ontology as it is not feasible to have utterances for all slot type-value combinations. It is also good to have multiple utterances for the same input context, which are often paraphrased version of each other. This can help a latent space generative model to learn to paraphrase which is common in human conversation. People rarely speak the same sequence of words even when the same person wants to convey the same thing again, or have the same input context.

Since rare slot-value pairs in training data is a problem, some methods use **delexicalization** as a preprocessing step before training a response generation model. Delexicalization involves substituting the slot values in the utterance by predefined generic placeholder tokens. To illustrate delexicalization, we can take an example from Figure 1.2. In turn 8, *"Red Joan"* in the system utterance *"Would you like to see the movie Red Joan?"* can be replaced by a token like *"[movie_name]"* giving *"Would you like to see the movie [movie_name]?"*. The response generation model can be trained to output the delexicalized utterance. **Relexicalization** is used to further process the generated utterance from the model, before giving it to the TTS model. Relexicalization is the process of replacing the generic tokens by specific slot values which can be obtained from the system act. This technique is especially useful for slot types like "name" and "phone numbers" because it is not practical to contain all possible values in both the domain ontology and the labeled data. However, you need to label the starting and ending location of the slot value in the utterance, unlike just filling in the slot value, which may be more tedious.

1.2 Thesis Outline

We described the fundamentals of dialog systems and the architectures in this chapter. In this dissertation, our contribution is two fold. Firstly, we provide a new approach for dialog state tracking which utilizes the dialog history. Secondly, we study deep generative models for system response generation for the task-oriented dialog setting. Following is an outline for the rest of the chapters:

1. **Chapter 2** covers the background and related work. It is divided into two parts: dialog state tracking and dialog generation.

2. **Chapter 3** presents our contribution on a novel method for dialog state tracking.

3. **Chapter 4** presents our study of two generative models, transformer encoder-decoder and conditional variational autoencoder, for task oriented dialog generation.

4. **Chapter 5** summarizes our contributions and discusses future work.

Turn	User	System	Dialog State / System Act
1	Could you help me with a restaurant reservation?		**service:** restaurant, **intent:** reserve restaurant
2		In what city and at what restaurant?	**request:** [city, restaurant_name]
3	The restaurant is Everest in Albany.		**service:** restaurant, **intent:** reserve restaurant **restaurant:** { **city:** Albany, **name:** Everest }
4		What time would you like the reservation?	**request:** [time]
5	I'd like it at 8 in the evening.		**service:** restaurant, **intent:** reserve restaurant **restaurant:** { **city:** Albany, **name:** Everest, **time:** 8pm }
6		Confirming a table for 2 at Everest Kitchen in Albany at 8 pm on March 4th.	**confirm:** [(**city:** Albany), (**time:** 8pm), (**party_size:** 2), (**date:** March 4th)]
7	Thank you for that. I'd like to see a movie, too. I'm interested in a drama at the Albany Twin, a regular show.		**service:** movie, **intent:** find movies **movie:** { **genre:** drame, **location:** Albany, **theater_name:** Albany Twin }
8		Would you like to see the movie Red Joan?	**offer:** [(**movie_name:** Red Joan)]
9	Sure, what time is the movie next Friday?		**service:** movie, **intent:** get times for movie **movie:** { **genre:** drame, **location:** Albany, **theater_name:** Albany Twin }
10		I have 1 showing at 7 PM at Landmark's Albany Twin.	**offer:** [(**show_time:** 7pm), (**theater_name:** Landmark's Albany Twin)] **Inform_count:** [(**count:** 1)]
11	That's everything I need, thank you!		**service:** movie, **intent:** get times for movie **movie:** { **genre:** drame, **location:** Albany, **theater_name:** Albany Twin }
12		You're very welcome.	**goodbye:** [ ]

Figure 1.2: A dialog sample from Schema-Guided DST dataset (2) spanning two domains (services): restaurant and movie. The first column shows the turn number where the user (yellow boxes) and system (red boxes) alternate taking turns. Green boxes show the cumulative dialog state for each user utterance. Restaurant service in the dialog state is omitted after turn 7 to save space. Blue boxes are the system acts corresponding to each system utterance.

2. BACKGROUND AND RELATED WORK

2.1 Dialog State Tracking

Early work for DST relied on separate Spoken Language Understanding (SLU) module (16) to extract relevant information from user utterances in a pipelined approach. Such systems are prone to error accumulation from a separate SLU module, in absence of necessary dialog context required to interpret the user utterance. Thus, later work on DST moved away from separate SLU modules and inferred the dialog state directly from user utterance and dialog history (17; 18; 19). These models depend on delexicalization, using generic tags to replace specific slot types and values, and handcrafted semantic dictionaries. In practice, it is difficult to scale these models for every slot type and recent state-of-the-art models for DST use deep learning based methods to learn general representations for user and system utterances and previous system actions, and predict the turn state (20; 17; 21; 22; 23; 24; 25; 26). However, these systems are found to perform poorly on rare and unknown slot-value pairs which was recently addressed through local slot-specific encoders (27) and pointer network (28).

A crucial limitation to all these approaches lies in the modeling of appropriate historical context, which is simply ignored in most of the works. Since user's goal may change back-and-forth between previous values, incorporating relevant historical context is useful in monitoring implicit goal references. In a recent work, (29) discussed on similar limitations of current DST task and introduced a new task of frame tracking that explicitly tracks every slot-values that were introduced during the dialogue. However, that significantly complicates the task by maintaining multiple redundant frames that are often left unreferenced. Our proposed model, that explicitly track relevant historical user and system utterances, can be easily incorporated into any known DST or frame tracking systems such as (30) to replace the recency encoding.

2.2 Dialog Generation

Encoder-decoder models, also known as seq2seq (sequence to sequence) models, (31; 32; 10) are the most common approach for corpus-based learning for context sensitive conversational response generation. However, a few modifications are required to these autoregressive models to obtain better performance on conversation response generation. One of the problems is that the model learns to predict repetitive and dull responses like *"I don't know"* or *"I'm fine."*. (33) added a mutual information objective in the training objective and modified the beam decoder for more diverse responses. (34) encode utterances and generate responses by incorporating different types of situation behind conversations. (35) proposed to combine topic-information with seq2seq models to generate informative responses. (36) augmented the encoder-decoder models with meta word, a structured record to describe various attributes in the response, for open domain dialog response generation.

Recent works in self-supervised **pre-training** like BERT (37) and **language models** like GPT-2 (38) have also been utilized for response generation in dialog. (39) used a BERT based model to rank the candidate responses with dot product between the context and candidate embeddings. TransferTransfo framework (40) proposed a model similar to GPT-2 transformer decoder and trained it on Persona-chat dataset (41). (42) built on the TransferTransfo framework by utilizing the transformer decoder for multi-domain task-oriented dialog generation. (43) also showed the effectiveness of generative pre-trained transformer model for open-domain dialog response generation.

More recently, (44) trained a large encoder-decoder evolved transformer (45) chatbot, called Meena, with a multi-turn (maximum 7, instead of a single last turn) context to minimize the response perplexity. Meena was significantly better than the other prior models trained on a single last turn context and exhibited characteristics of human conversation like consistency automatically. (46) used the self-attention mechanism to find the relevant context from the **dialog history** for multi-turn dialog generation. (47) model the dialog history with finite state transducer for non-collaborative conversations which require not only the semantic intent but also the tactics that convey the intent. (48) utilize the order information in dialog history for self-supervised learning on open domain dialog generation.

Multiple responses may be appropriate for a same dialog context. There have been some recent work which try to address this **one-to-many** property of response generation in conversations. (49) proposed Multi-action data augmentation (MADA) technique which maps current dialog state to different valid system actions to generate diverse system responses. (50) presented a technique that combined supervised learning and reinforcement learning to generate multiple responses in a task-oriented dialog setting.

Latent variable models (51) are also very popular for response generation. After the success of Variational autoencoders (VAE) (52) for text generation (53; 54; 55), recent works have used VAE for dialog response generation (56; 57). Conditional variational autoencoder (CVAE) (6) is a variant of VAE which has also been utilized for dialog generation (58). A common problem with these latent variable model is posterior collapse where the decoder learns to ignore the latent space vector and the encoder. Several recent works have tried to address this problem (59; 60). (56) learn a discrete sentence representation for interpretable dialog response generation. (61) use VAE to extract dialog structure via unsupervised learning. (62) presented a conditional Wasserstein autoencoder (DialogWAE) which modeled the distribution of data by training a GAN (63) within the latent variable space.

Previous works have also generated dialog responses with cross-domain latent actions via zero-shot learning (64). (65) take intuition from turing test and cast the problem of dialog generation as reinforcement learning where they do adversarial training of two dialog agents. (66) trained a chatbot using policy gradient to reward sequences that showed three useful properties, namely, informativity, coherence, and easy of answering.

3. DISCERNING RELEVANT CONTEXT FOR DST *

Narrow domain goal oriented dialog systems have numerous real world applications in fields like e-commerce, health-care, education, customer care and video-games. These systems function by accessing or acting upon an external knowledge base by using a structured application program interface (API). The API calls require some set of arguments which are stored in the belief state of dialog system, often called dialog state. Dialog state tracking (DST) is a vital component in the task-oriented dialog systems which is used to estimate user's goals and requests in order to plan next action and respond accordingly. At each turn, DST aims to identify the set of goals that a user aims to achieve and requests that are represented as slot-value pairs. Typically, this decision is made by considering user utterance in the current turn or system actions in the previous turn. However, in many cases, the considered user utterance or system actions do not present enough information and refers to a previous utterance.

As shown through an example in Figure 3.1, while exploring different available options, user can go back-and-forth between the currently and previously discussed facts. For instance, when offered with two different restaurant options namely *Nirala* `(food=indian)` and *Golden Wok* `(food=chinese)` in the second turn, user first inquires about the details of *Golden Wok*. And after getting relevant details about the *Golden Wok* in the following two turns, user refers back to the second option provided in second turn and asks about *Nirala* restaurant. To predict the correct slot-value pair `food=indian` in the dialog state of the fifth turn, the system is required to refer back to the second turn again to find information about *Nirala*, as the context obtained from the current dialog turn is insufficient.

Identifying such implicitly referenced historical turns is challenging since implicit references are not local and most recent turns are often not informative. Therefore, the traditional approach of

*Reprinted with permission from "Improving Dialogue State Tracking by Discerning the Relevant Context" by Sanuj Sharma, Prafulla Kumar Choubey, Ruihong Huang, 2019 (4). 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). 576–581. Copyright © 1963–2019 ACL. .

User: hello, i'm looking for a restaurant with fair prices
System: There are 31 places with moderate price range. Can you please tell me what kind of food you would like?
Turn Label: price range = moderate **Sys Act:** food

User: well I want to eat in the North, what's up that way?
System: I have two options that fit that description, Golden Wok chinese restaurant and Nirala which serves Indian food. Do you have a preference?
Turn Label: area = north **Sys Act:** food

User: Can I have the address and phone number for the Golden Wok chinese restaurant?
System: The phone number is 01223 350688.
Turn Label: request = address, phone; food = chinese

User: thank you. what is the address?
System: The address is 191 Histon Road Chesterton.
Turn Label: request = address

User: Okay, what about Nirala, what's the address and phone of that?
System: 7 Milton Road Chesterton and the number is 01223 360966
Turn Label: request = address, phone; food = indian

Figure 3.1: An example dialog from WoZ 2.0 dataset (3). A turn contains user utterance (blue), system utterance (red), system actions (yellow) and turn label (green). Each turn is separated by a line. Reprinted with permission from (4).

modeling dialogue recency (29) may not suffice. Instead, we propose to model implicit references by storing links to the past turn where each of the slots was modified. Then at each turn, we look up though the stored links to find the previous turn which may provide additional cues for predicting the appropriate slot-value.

Moreover, the dialogue system often asks polar questions with yes-no answers. For instance, the DST system should update the dialogue state with `food=indian` when a user replies *Yes* to a system utterance *Do you want Indian food?*. In such cases, neither the user utterance nor system acts (*food* in this example) contain any information about the actual slot-value. This makes

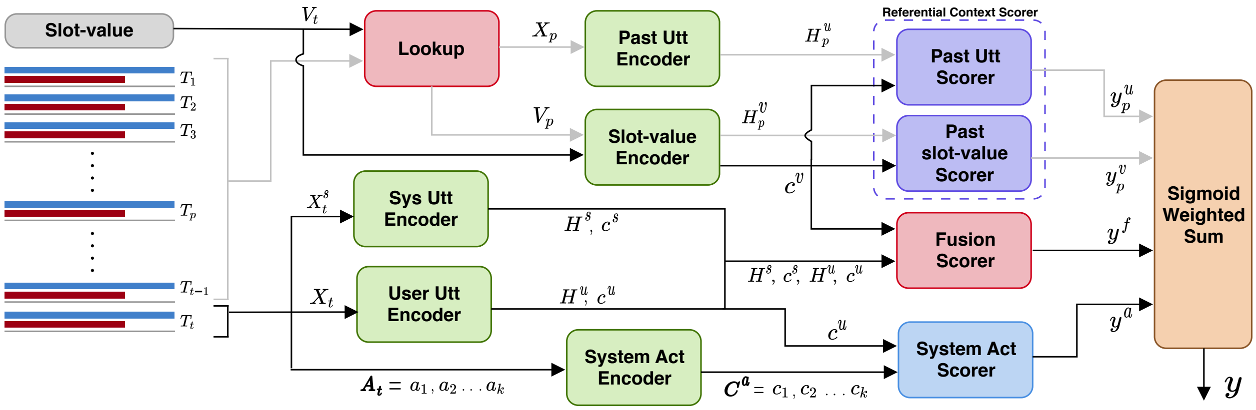

Figure 3.2: The Architecture of Context Aware Dialogue State Tracker. Reprinted with permission from (4).

utilization of both system and user utterance eminent for dialog state tracking. However, utilizing the previous system utterance together with the current user utterance always at each turn may add noise. Therefore, we use a gating mechanism based on both utterances to determine the relevance of the previous system utterance in the current turn.

The evaluation shows that identifying the relevant context is essential for dialogue state tracking. Our novel model that discerns important details in non-adjacent dialogue turns and the previous system utterance from a dialog history is able to improve the previous state-of-the-art GLAD (27) model on all evaluation metrics for both WoZ and MultiWoZ (restaurant) datasets. Furthermore, we empirically show that a simple self-attention based biLSTM model, using only one-third of the number of parameters as GLAD, outperforms GLAD by identifying and incorporating the relevant context.

3.1 Model

Similar to previous works, we decompose the multi-label classification problem to binary classification where we score each slot-value pair and select the ones that receive a score above a threshold to be included in the current dialog state. To predict the score for a candidate slot-value pair, the model uses the relevant past user utterance (referential utterance), a fused utterance composed using the current user utterance and the system utterance of the previous turn, as well as previous system actions as evidence. Shown in Figure 3.2, our model comprises of:

3.1.1 Lookup module

The Lookup module retrieves a link to the turn where each of the slots changes. At each step, our system refers to the lookup module that returns the past user utterance (the "*antecedent user utterance*") where the candidate slot-type was modified as well as outputs the previous slot-value.

3.1.2 GLE module

Each of the five green modules in Figure 3.2 is a global-locally self-attentive encoder (GLE module) (27) that encodes each type of evidence into a vector representation (c). Each input is represented as a sequence of words which is encoded to a vector representation via global-local self-attentive encoder (GLE) module (27). Specifically, GLE employs local slot-specific bidirectional LSTMs and a global bidirectional LSTM (67) that is shared across all slots for encoding the input sequence into a sequence of hidden states (H), followed by a self-attention layer (68) to obtain a fixed dimension vector representation (c).

The GLE modules are used to encode the antecedent user utterance (H_p^u, c_p^u), the current user utterance (H^u, c^u), the previous system utterance (H^s, c^s), each of the system acts (H^{a_i}, c^{a_i}), as well as the previous slot-value (H_p^v, c_p^v) and the candidate (H^v, c^v) slot-value.

3.1.3 Referential Context Scorer

The Referential Context Scorer uses the candidate slot value (c^v), the antecedent user utterance as well as the previous slot-value to determine if the candidate slot value was referenced in the antecedent utterance. Specifically, the scorer uses the representation of the candidate slot value c^v to attend over hidden states of the antecedent user utterance and the previous slot-value, H_p^u and H_p^v, and then computes attention weights for each of the hidden states. Next, the scorer sums up the hidden states weighed with the calculated attentions to get the summary context (Equation 3.1). Finally, the scorer applies a linear neural layer to calculate the scores y_p^v and y_p^u representing the likelihoods that the candidate slot-value is different from the previous slot-value and the candidate slot-value was unreferenced in the antecedent utterance (Equation 3.2).

$$
\begin{aligned}
Q(H,c):\ a_j = (H_j)^\top c\ ;\ p = softmax(a) \\
Q(H,c) = \sum_i p_i H_i
\end{aligned}
\tag{3.1}
$$

$$
\begin{aligned}
y_p^u = W_p^u\, Q(H_p^u, c^v) + b_p^u \\
y_p^v = W_p^v\, Q(H_p^v, c^v) + b_p^v
\end{aligned}
\tag{3.2}
$$

3.1.4 Fusion Scorer

This scorer leverages necessary details in the previous system utterance to enrich the current user utterance. First, we use a gating mechanism based on c^s and c^u that determines the relevance of the previous system utterance in the current turn. We concatenate c^s and c^u and use a linear layer with sigmoid activation to calculate the score α (Equation 3.3). Then, we use attention from c^v over H^s and H^u to calculate context summaries (l^s, l^u), and combine the summary vectors by taking their normalized weighted sum based on α. We finally apply a single linear layer to calculate the score y^f that determines the likelihood of the candidate slot-value based on both the current user utterance and the previous system utterance (Equation 3.4).

$$
\begin{aligned}
f_c = W_{fc}(c^s \oplus c^u) + b_{fc} \\
\alpha = \sigma(W_\alpha tanh(f_c) + b_\alpha)
\end{aligned}
\tag{3.3}
$$

$$
\begin{aligned}
l^s = Q(H^s, c^v)\ ;\ l^u = Q(H^u, c^v) \\
l^f = \alpha l^s + (1-\alpha) l^u\ ;\ y^f = W_{lf} l^f + b_{lf}
\end{aligned}
\tag{3.4}
$$

3.1.5 System Act Scorer

The System Act Scorer is the same as the action scorer proposed by (27). Specifically, The scorer uses attention from c^u over C^a to calculate action summary followed by a linear layer with sigmoid activation to calculate the score y^a that determines the relevance of the candidate slot-value based on the previous system actions (Equation 3.5).

$$l^a = Q(C^a, c^u)\,; \quad y^a = (l^a)^\top c^v \tag{3.5}$$

It then calculates the final score of the candidate slot-value by taking weighted sum of the four scores (y_p^u, y_p^v, y^f, y^a) followed by a sigmoid layer, where weights are learned in the network.

3.2 Evaluations

The following setup was used for conducting empirical analysis for the model described in Section 3. However, similar setup is also applicable for the sparse factor graph model.

3.2.1 Experimental Setup

We primarily use WoZ 2.0 (3) restaurant reservation task dataset that consists of 1200 dialogues for training and evaluation. Each dialogue has an average of eight turns, where each turn contains *system utterance transcript*, *user utterance transcript*, *turn label* and *belief state*. All the dialogue states and actions are based on a task ontology that supports three different informable slot-types namely *price range* with 4 values, *food* with 72 values, *area* with 7 values, and *requests* of 7 different types like *address* and *phone*. Following the standard settings, we use 600 dialogues for training, 200 for validation and the remaining 400 for testing.

We also use dialogues from restaurant domain in MultiWoZ 2.0 dataset (7) for secondary evaluation. It banks on a significantly complex ontology covering seven informable slot types with 276 different values (*food, price range, restaurant name, area, book time, book day* and *book people* with 97, 6, 105, 8, 43, 8 and 9 values respectively). We use standard training, validation and test splits of 1199, 50 and 61 dialogues respectively.

All the models on WoZ 2.0 are evaluated on the two standard metrics introduced in (69). First, **Joint Goal Accuracy** is the percentage of turns in a dialogue where the user's informed joint goals are identified correctly. Joint goals are accumulated turn goals up to the current dialog turn. Second, **Turn Request Accuracy** calculates the percentage of turns in a dialogue where the user's

Model	WoZ 2.0 Joint Goal	 Turn Request
Delexalisation-Based Model + SD	83.7%	87.6%
NBT - DNN	84.4%	91.2%
NBT - CNN	84.2%	91.6%
GLAD †	**86.4%**	**97.1%**
Global biLSTM based GLE	85.0%	96.8%
Global biLSTM based GLE + RC	87.4%	**97.0%**
Global biLSTM based GLE + RC + FS	**88.4%**	**97.0%**
GLAD + RC + FS	**89.2%**	**97.4%**

Table 3.1: Test accuracy of baselines and proposed models on WoZ 2.0 restaurant reservation dataset. Reprinted with permission from (4).

requests were correctly identified. Models on MultiWoZ 2.0 dataset are evaluated using joint goal and turn inform accuracies, as used by (70).

3.2.2 Implementation Details

We use pretrained GloVe word embeddings (71) concatenated with character n-gram embeddings (72) which are kept fixed during the training. Each of bi-LSTMs use 200 hidden dimensions. All the models are trained using ADAM optimizer (73) with the initial learning rate of 0.001. Dropout rate (74) is set to 0.2 for all biLSTM modules and the embedding layer. The models are trained for a maximum of 100 epochs with a batch size of 50. The validation data was used for early stopping and hyperparameter tuning.

3.2.3 Results

Table 3.1 compares the performance of our proposed models with different baselines, including **delexalisation-based model + SD** (3), DNN and CNN variants of **neural belief tracker** (22) and the previous state-of-the-art GLAD systems (27) on WoZ 2.0 dataset. We also implement a simplified variant of GLAD, **Global BiLSTM based GLE**, by removing slot-specific local biLSTMs

Model	MultiWoZ 2.0 (Restaurant) Joint Goal	Turn Inform
GLAD	43.95%	76.99%
GLAD + RC	**45.72%**	**77.87%**
GLAD + RC + FS	**46.31%**	**78.76%**

Table 3.2: Test accuracy of GLAD and proposed models on MultiWoZ 2.0 restaurant domain dataset. Note that we considered all 276 slot-values for evaluating models. (7) reported joint goal accuracy of 80.9 on MultiWoZ 2.0 (restaurant) dataset. We believe they didn't include *restaurant name* slot in their evaluation and only considered presence of three slot-types—*book time, book day* and *book people*—and not their values. Reprinted with permission from (4).

from the GLE encoder. We then successively combine it with referential context (**Global biLSTM based GLE + RC**) and the fused previous system utterance (**Global biLSTM based GLE + RC + FS**). Finally, we directly incorporate the referential context and gate selected system utterance into the GLAD system (**GLAD + RC + FS**).

Irrespective of the underlying system, utilizing appropriate context from the previous turns improves the overall performance of a dialogue state tracker on both joint goal and turn request accuracies on WoZ 2.0 dataset. First, incorporating relevant referential utterances to identify implicitly mentioned slot-value improves the accuracy of global biLSTM based GLE model on joint goal task by 2.4%. Then, gating based mechanism to augment user utterance with relevant information from the previous system utterance further improves the joint goal accuracy by 1.0%. Together, they improve joint goal and request accuracy of the global biLSTM based GLE model by 3.4% and 0.2% respectively. Furthermore, as evident from the results in Table 3.2, both referential context and fused system utterance proportionally improve performance on MultiWoZ 2.0 dataset as well with overall improvement of 2.36% and 1.77% on joint goal and turn inform accuracies respectively. Performances of all models on MultiWoZ 2.0 are significantly inferior compared to WoZ 2.0 owing to higher complexity, with richer and longer utterances and considerably more

Model	Approx. # of parameters
Global biLSTM based GLE	1.2 million
Global biLSTM based GLE + RC + FS	**6 million**
GLAD	17 million
GLAD + RC + FS	28 million

Table 3.3: Number of learnable parameters for different models on WoZ 2.0 dataset. Reprinted with permission from (4).

slot-values in the former dataset.

3.3 Analysis and Discussion

The utilization of relevant context results in significant reduction in the number of learnable parameters in the model as shown in Table 3.3. Relevant context with the baseline model is able to outperform GLAD while using only one third of the number of learnable parameters. The parameters added due to using relevant context are the parameters for encoding the antecedent referential user utterance and the previous system utterance as well as the past utterance and past slot-value scorers. However, we also observe high variance in the joint goal accuracy. Since joint goal is calculated by accumulating turn goals, an error in predicting a turn goal is propagated to all the downstream turns.

3.4 Conclusion

We have proposed a novel method for identifying the relevant historical user utterance as well as determining the relevance of the system utterance from the last turn to enrich the current user utterance and improve goal tracking in dialogue systems. Specifically, we use a mixture of user utterance and system utterance fused together by their relative importance and the utterance and predicted slot-value pair from a past turn where the slot type under consideration was last modified. The experimental results show that the extracted context is able to capture details, from both user and system utterance, discerning relevant context from the dialog history is crucial for tracking dialog states. More importantly, the previous turn utterances allow the system to refer back to prior turns in dialog history, which helps in implicit reference resolution. By using the proposed relevant context, we are able to achieve joint goal accuracy of 89.2% and 97.3% turn request accuracy on WoZ 2.0 data-set. We also conducted experiments on restaurant domain of MultiWoz dataset to achieve joint goal accuracy of 46.31% and turn inform accuracy of 76.76%.

4. DIALOG CONTEXT TO TEXT

In this chapter, we talk about dialog generation for task-oriented dialog agents. One of the problems that we address in this work is the one-to-many problem in dialog generation. There are multiple responses possible for the same dialog context comprising of the dialog state, last user utterance and the system act. For instance, in turn 2 in Figure 1.2 the system could also have responded by saying *"Which city do you live in? And, do you have any particular restaurant in mind?"* instead of *"In what city and at what restaurant?"*.

4.1 Approach

We study two generative models for the task of dialog generation. The first model is a transformer encoder-decoder model (5) which uses multiple layers attention mechanism in both the encoding and decoding process. It uses scaled dot-product attention which can be described as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4.1}$$

Here Q is a matrix containing a set of queries, K and V are metrics for keys and values. Unlike an RNN which processes the text in sequential manner, a transformer model processes the text in feed-forward manner. The input to the transformer model comprises of the dialog state, last user utterance and the system act, which are concatenated together separated by special tokens from the vocabulary.

The second generative model is a conditional variational encoder (CVAE) (6). Dialog generation can be defined as a conditional distribution:

$$p(x, z|c) = p(x|z, c)p(z|c) \tag{4.2}$$

Here, x is the system response, z is a latent variable, and c is the context comprising of the dialog state, last user utterance and the system act. $p(x|z, c)$ and $p(z|c)$ are parametrized by neural networks with parameters θ. As shown in figure 4.1, $p_\theta(z|c)$ is the prior network and $p_\theta(x|z, c)$ is the response decoder. Then to sample a dialog response, first a latent variable z is sampled from $p_\theta(z|c)$, and then x is generated through the response decoder $p_\theta(x|z, c)$. The true posterior $p(z|x, c)$ is approximated by a recognition network $q_\phi(z|x, c)$ where there is an assumption that z follows multivariate Gaussian distribution with diagonal covariance matrix. The lower bound can be written as:

$$L(\theta, \phi; x, c) = E_{q_\phi(z|c,x)}[log p_\theta(x|z, c)] - KL(q_\phi(z|x, c)||p_\theta(z|c)) \tag{4.3}$$

The network is trained with the stochastic gradient variational bayes framework (52) to maximize the above lower bound.

The advantage of the CVAE is that, unlike transformer encoder-decoder model, we can use the latent variable z to sample different responses for the same dialog input context.

4.2 Dataset

We use the schema-guided dialog state tracking dataset (2) for empirical analysis of our models. The dataset consists of conversations between a virtual assistant and users. The data collection setup used a dialog simulator to generate dialog outlines first and then paraphrasing was done via crowdsourcing to obtain natural utterances. The dataset contains over 16000 dialogues in the training set with 16 different services (domains). Some examples of services are banks, calendar, flights, hotels, movies, restaurants etc. Each dialog has an average of 20 turns. The dialog ontology contains 214 slot types and over 14000 slot values.

4.3 Results

We use **perplexity** as a metric to evaluate both of the generative models.

$$PP(x) = P(x_1 x_2 x_3 ... x_n)^{1/n} \tag{4.4}$$

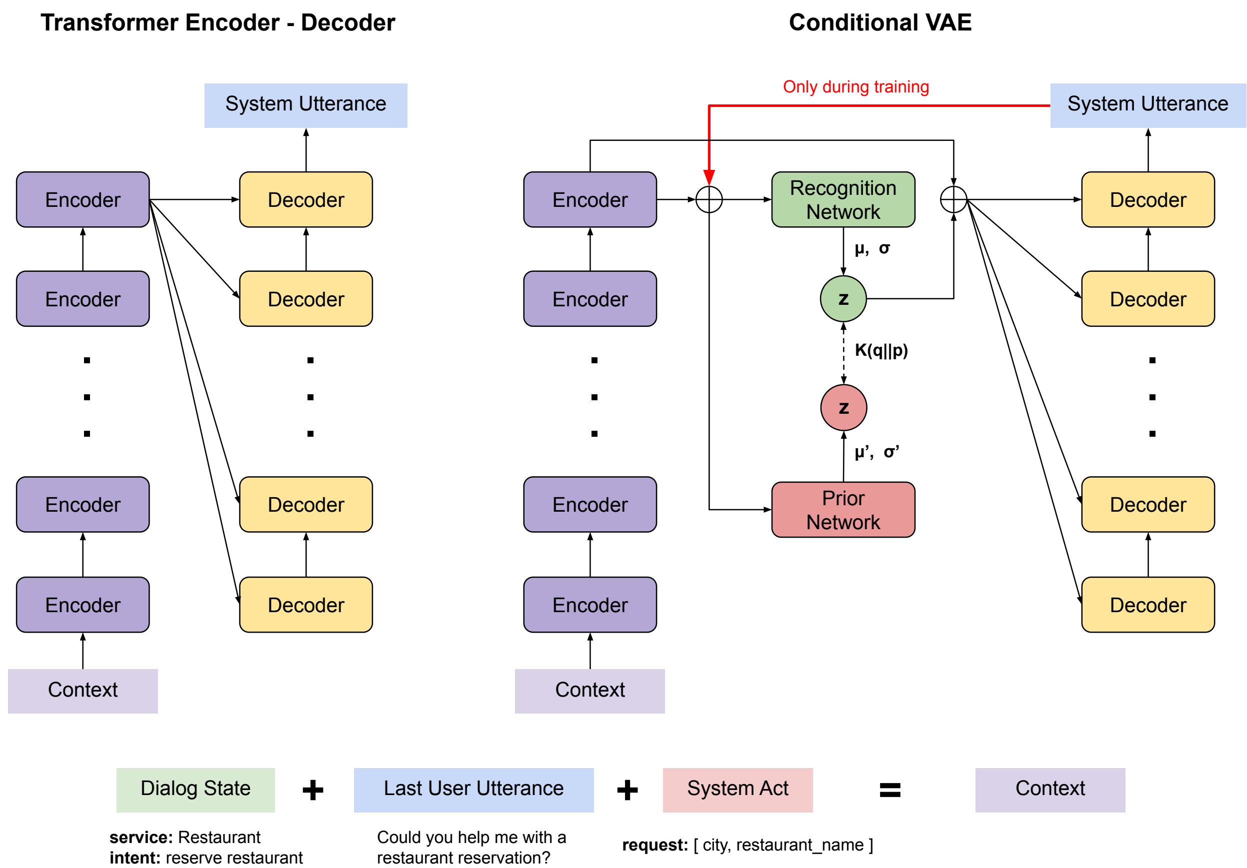

Figure 4.1: The dialog state, last user utterance, and system act are concatenated, separated by special tokens, to form the input context, c, which is fed to the response generation model. On the left is a transformer encoder-decoder model (5) which takes in the input context and outputs the system response. On the right is a CVAE (6) which has an encoding and and decoding process similar to the transformer model, but also has a recognition network for $q_\phi(z|x,c)$ and a prior network for $p_\theta(z|c)$. The recognition network is only used during training and for testing z is sampled from the prior network.

Table 4.1 shows the perplexity score for the transformer encoder-decoder and the conditional variational auto-encoder. The models were trained for 10 epochs each. Greedy decoding was used for the transformer for generating responses. The perplexity score for CVAE was higher than that of the transformer. One of the reasons may be because of the need for sampling from the prior distribution of the latent variable z. For calculating perplexity, z was sampled three times and the reported score is an average of those three samples. Like the decoding in the transformer model, the CVAE also utilized greedy decoding for generating responses.

Model	Perplexity
Transformer encoder-decoder	46.7
Conditional VAE	48.3

Table 4.1: Perplexity scores for the transformer encoder-decoder and CVAE on schema-guided dialog state tracking dataset.

4.4 Discussion

As mentioned in the previous section, the CVAE model scores a higher perplexity than the transformer model. However, there is an advantage to the CVAE that we want to highlight. Since CVAE has a latent space z, it is possible to sample from the latent space distribution. This ability to sample in the latent space allows the generated system response to be a paraphrased version of the system response for the input context. This is not possible in case of the transformer model. The transformer outputs the same system response for a single input context. The only way to change the output of the transformer is by changing the decoding procedure, which is also possible in the CVAE decoder. Prior works have utilized a CVAE to model the discourse level characteristics with the latent space. However, here all the discourse level information is given in the input context to the model, and the model just learns to paraphrase. Since modeling discourse level characteristics is a much complex problem than paraphrasing, the latent space is quite effective in paraphrasing the system response. Furthermore, it is pretty resourceful to have the paraphrasing ability in the system response generation model since the response is not always the same for a fixed input context in a conversation.

4.5 Conclusion

In this chapter, we studied two generative models, namely, transformer and CVAE, for learning to generate system response from the input context. The input context to the model was composed of the dialog state, last user utterance and the system act. We used perplexity to compare the two

Input: <|START|> **i am feeling hungry so i would like to find a place to eat .**<|END|> <|ACTSTART|> **request city** <|ACTEND|>

Output: <|START|> **what city do you want to look in ?** <|END|>

Actual: <|START|> **do you have a specific which you want the eating place to be located at ?** <|END|>

Input: <|SVSTART|> **city is san jose** <|SVEND|> <|START|> **i would like for it to be in san jose .**<|END|> <|ACTSTART|> **request cuisine is mexican italian** <|ACTEND|>

Output: <|START|> **what kind of food do you like ? mexican , italian , or something else ?** <|END|>

Actual: <|START|> **is there a specific cuisine type you enjoy , such as mexican , italian or something else ?** <|END|>

Input: <|SVSTART|> **city is san jose cuisine is american** <|SVEND|> <|START|> **is there some other restaurant which you can suggest ?** <|END|> <|ACTSTART|> **offer restaurant name is bazille offer city is san jose** <|ACTEND|>

Output: <|START|> **in san jose i have found a nice restaurant called bazille restaurant .** <|END|>

Actual: <|START|> **how would you like bazille restaurant which is situated in san jose .** <|END|>

Figure 4.2: System response samples from the transformer encoder-decoder.

Input: <|SVSTART|> **city is san jose cuisine is american** <|SVEND|><|START|> **is there some other restaurant which you can suggest ?** <|END|> <|ACTSTART|> **offer restaurant name is bazille offer city is san jose** <|ACTEND|>

Output 1: <|START|> **in san jose i have found a nice restaurant called bazille restaurant .** <|END|>

Output 2: <|START|> **there is a restaurant in san jose called bazille restaurant .** <|END|>

Output 3: <|START|> **you can like bazille restaurant in san jose.** <|END|>

Actual: <|START|> **how would you like bazille restaurant which is situated in san jose .** <|END|>

Input: <|SVSTART|> **city is palo alto cuisine is american price range is moderate restaurant name is bird dog** <|SVEND|> <|START|> **alright , that seems good . i would like to make a booking at this restaurant .** <|END|> <|ACTSTART|> **request time** <|ACTEND|>

Output 1: <|START|> **can you tell me the time when you want to book the table for ?** <|END|>

Output 2: <|START|> **i require a time before booking a table .** <|END|>

Output 3: <|START|> **when is your booking table ?** <|END|>

Actual: <|START|> **how would you like bazille restaurant which is situated in san jose .** <|END|>

Figure 4.3: System response samples from the conditional variational auto-encoder.

models and found that both the models have comparable performance. Furthermore, CVAE allows the ability to paraphrase the system response which is not possible in the transformer model.

5. CONCLUSION

In this work, we proposed a novel technique for dialog state tracking which uses heuristics to pick relevant turns from the dialog history (4). We showed how this heuristics based approach can be applied with LSTM and self-attention encoders to improve the DST performance on two datasets, namely, WoZ 2.0 and MultiWoZ 2.0 (Restaurant domain). We also presented a new way to combine user and system utterances with a weighted sum to obtain a single vector representation for the dialog turn. Combining system and user utterance, and utilizing that for DST further improved the performance of the model.

Secondly, we studied two generative models for dialog generation in a task-oriented setting. More specifically, we used transformer encoder-decoder and CVAE for generating system responses in schema-guided DST dataset.

In future, we plan to extend the latent space models like CVAE for dialog generation with normalizing flows (75).

REFERENCES

[1] J. Williams, A. Raux, and M. Henderson, "The dialog state tracking challenge series: A review," *Dialogue & Discourse*, vol. 7, no. 3, pp. 4–33, 2016.

[2] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, "Schema-guided dialogue state tracking task at dstc8," *arXiv preprint arXiv:2002.01359*, 2020.

[3] T.-H. Wen, D. Vandyke, N. Mrkšić, M. Gasic, L. M. Rojas Barahona, P.-H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 438–449, Association for Computational Linguistics, 2017.

[4] S. Sharma, P. K. Choubey, and R. Huang, "Improving dialogue state tracking by discerning the relevant context," *arXiv preprint arXiv:1904.02800*, 2019.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, Curran Associates, Inc., 2017.

[6] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, pp. 3483–3491, 2015.

[7] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic, "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*, pp. 5016–5026, 2018.

[8] L. Wittgenstein, *Philosophical investigations*. John Wiley & Sons, 2009.

[9] J. L. Austin, *How to do things with words*, vol. 88. Oxford university press, 1975.

[10] O. Vinyals and Q. Le, "A neural conversational model," *arXiv preprint arXiv:1506.05869*,

2015.

[11] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine," *Communications of the ACM*, vol. 9, no. 1, pp. 36–45, 1966.

[12] K. M. Colby, S. Weber, and F. D. Hilf, "Artificial paranoia," *Artificial Intelligence*, vol. 2, no. 1, pp. 1–25, 1971.

[13] A. M. Turing, "Computing machinery and intelligence," in *Parsing the Turing Test*, pp. 23–65, Springer, 2009.

[14] D. G. Bobrow, R. M. Kaplan, M. Kay, D. A. Norman, H. Thompson, and T. Winograd, "Gus, a frame-driven dialog system," *Artificial intelligence*, vol. 8, no. 2, pp. 155–173, 1977.

[15] M. Fazel-Zarandi, S.-W. Li, J. Cao, J. Casale, P. Henderson, D. Whitney, and A. Geramifard, "Learning robust dialog policies in noisy environments," *arXiv preprint arXiv:1712.04034*, 2017.

[16] M. Henderson, M. Gašić, B. Thomson, P. Tsiakoulis, K. Yu, and S. Young, "Discriminative spoken language understanding using word confusion networks," in *Spoken Language Technology Workshop (SLT), 2012 IEEE*, pp. 176–181, IEEE, 2012.

[17] M. Henderson, B. Thomson, and S. Young, "Word-based dialog state tracking with recurrent neural networks," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 292–299, Association for Computational Linguistics, 2014.

[18] M. Henderson, B. Thomson, and S. Young, "Word-based dialog state tracking with recurrent neural networks," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 292–299, 2014.

[19] L. Zilka and F. Jurcicek, "Incremental lstm-based dialog state tracker," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*, pp. 757–762, IEEE, 2015.

[20] M. Henderson, B. Thomson, and S. Young, "Deep neural network approach for the dialog state tracking challenge," in *Proceedings of the SIGDIAL 2013 Conference*, pp. 467–471,

Association for Computational Linguistics, 2013.

[21] N. Mrkšić, D. Ó Séaghdha, B. Thomson, M. Gasic, P.-H. Su, D. Vandyke, T.-H. Wen, and S. Young, "Multi-domain dialog state tracking using recurrent neural networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 794–799, Association for Computational Linguistics, 2015.

[22] N. Mrkšić, D. Ó Séaghdha, T.-H. Wen, B. Thomson, and S. Young, "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)*, pp. 1777–1788, Association for Computational Linguistics, 2017.

[23] T. Hori, H. Wang, C. Hori, S. Watanabe, B. Harsham, J. Le Roux, J. R. Hershey, Y. Koji, Y. Jing, Z. Zhu, *et al.*, "Dialog state tracking with attention-based sequence-to-sequence learning," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 552–558, IEEE, 2016.

[24] B. Liu and I. Lane, "An end-to-end trainable neural network model with belief tracking for task-oriented dialog," *arXiv preprint arXiv:1708.05956*, 2017.

[25] F. Dernoncourt, J. Y. Lee, T. H. Bui, and H. H. Bui, "Robust dialog state tracking for large ontologies," in *Dialogues with Social Robots*, pp. 475–485, Springer, 2017.

[26] Y.-N. Chen, D. Hakkani-Tür, G. Tür, J. Gao, and L. Deng, "End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding.," in *Interspeech*, pp. 3245–3249, 2016.

[27] V. Zhong, C. Xiong, and R. Socher, "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)*, pp. 1458–1467, Association for Computational Linguistics, 2018.

[28] P. Xu and Q. Hu, "An end-to-end approach for handling unknown slot values in dialogue state tracking," in *Proceedings of the 56th Annual Meeting of the Association for Computa-*

tional Linguistics (Volume 1: Long Papers), pp. 1448–1457, Association for Computational Linguistics, 2018.

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

[30] H. Schulz, J. Zumer, L. E. Asri, and S. Sharma, "A frame tracking model for memory-enhanced dialogue systems," *arXiv preprint arXiv:1706.01690*, 2017.

[31] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," *arXiv preprint arXiv:1503.02364*, 2015.

[32] A. Sordoni, M. Galley, M. Auli, C. Brockett, Y. Ji, M. Mitchell, J.-Y. Nie, J. Gao, and B. Dolan, "A neural network approach to context-sensitive generation of conversational responses," *arXiv preprint arXiv:1506.06714*, 2015.

[33] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," *arXiv preprint arXiv:1510.03055*, 2015.

[34] S. Sato, N. Yoshinaga, M. Toyoda, and M. Kitsuregawa, "Modeling situations in neural chat bots," in *Proceedings of ACL 2017, Student Research Workshop*, (Vancouver, Canada), pp. 120–127, Association for Computational Linguistics, July 2017.

[35] C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, and W.-Y. Ma, "Topic aware neural response generation," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[36] C. Xu, W. Wu, C. Tao, H. Hu, M. Schuerman, and Y. Wang, "Neural response generation with meta-words," *arXiv preprint arXiv:1906.06050*, 2019.

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[38] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.

[39] J. Urbanek, A. Fan, S. Karamcheti, S. Jain, S. Humeau, E. Dinan, T. Rocktäschel, D. Kiela,

A. Szlam, and J. Weston, “Learning to speak and act in a fantasy text adventure game,” *arXiv preprint arXiv:1903.03094*, 2019.

[40] T. Wolf, V. Sanh, J. Chaumond, and C. Delangue, “Transfertransfo: A transfer learning approach for neural network based conversational agents,” *arXiv preprint arXiv:1901.08149*, 2019.

[41] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, “Personalizing dialogue agents: I have a dog, do you have pets too?,” *arXiv preprint arXiv:1801.07243*, 2018.

[42] P. Budzianowski and I. Vulić, “Hello, it’s gpt-2–how can i help you? towards the use of pretrained language models for task-oriented dialogue systems,” *arXiv preprint arXiv:1907.05774*, 2019.

[43] Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, and B. Dolan, “Dialogpt: Large-scale generative pre-training for conversational response generation,” *arXiv preprint arXiv:1911.00536*, 2019.

[44] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, *et al.*, “Towards a human-like open-domain chatbot,” *arXiv preprint arXiv:2001.09977*, 2020.

[45] D. R. So, C. Liang, and Q. V. Le, “The evolved transformer,” *arXiv preprint arXiv:1901.11117*, 2019.

[46] H. Zhang, Y. Lan, L. Pang, J. Guo, and X. Cheng, “Recosa: Detecting the relevant contexts with self-attention for multi-turn dialogue generation,” *arXiv preprint arXiv:1907.05339*, 2019.

[47] Y. Zhou, Y. Tsvetkov, A. W. Black, and Z. Yu, “Augmenting non-collaborative dialog systems with explicit semantic and strategic dialog history,” *arXiv preprint arXiv:1909.13425*, 2019.

[48] J. Wu, X. Wang, and W. Y. Wang, “Self-supervised dialogue learning,” *arXiv preprint arXiv:1907.00448*, 2019.

[49] Y. Zhang, Z. Ou, and Z. Yu, “Task-oriented dialog systems that consider multiple appropriate responses under the same context,” *arXiv preprint arXiv:1911.10484*, 2019.

[50] J. Rajendran, J. Ganhotra, S. Singh, and L. Polymenakos, "Learning end-to-end goal-oriented dialog with multiple answers," *arXiv preprint arXiv:1808.09996*, 2018.

[51] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio, "A hierarchical latent variable encoder-decoder model for generating dialogues," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[52] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[53] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[54] Y. Xiao, T. Zhao, and W. Y. Wang, "Dirichlet variational autoencoder for text modeling," *arXiv preprint arXiv:1811.00135*, 2018.

[55] L. Fang, C. Li, J. Gao, W. Dong, and C. Chen, "Implicit deep latent variable models for text generation," *arXiv preprint arXiv:1908.11527*, 2019.

[56] T. Zhao, K. Lee, and M. Eskenazi, "Unsupervised discrete sentence representation learning for interpretable neural dialog generation," *arXiv preprint arXiv:1804.08069*, 2018.

[57] X. Shen, H. Su, S. Niu, and V. Demberg, "Improving variational encoder-decoders in dialogue generation," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[58] T. Zhao, R. Zhao, and M. Eskenazi, "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders," *arXiv preprint arXiv:1703.10960*, 2017.

[59] Y. Park, J. Cho, and G. Kim, "A hierarchical latent structure for variational conversation modeling," *arXiv preprint arXiv:1804.03424*, 2018.

[60] J. Gao, W. Bi, X. Liu, J. Li, G. Zhou, and S. Shi, "A discrete cvae for response generation on short-text conversation," *arXiv preprint arXiv:1911.09845*, 2019.

[61] W. Shi, T. Zhao, and Z. Yu, "Unsupervised dialog structure learning," *arXiv preprint arXiv:1904.03736*, 2019.

[62] X. Gu, K. Cho, J.-W. Ha, and S. Kim, "Dialogwae: Multimodal response generation with conditional wasserstein auto-encoder," *arXiv preprint arXiv:1805.12352*, 2018.

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[64] T. Zhao and M. Eskenazi, "Zero-shot dialog generation with cross-domain latent actions," *arXiv preprint arXiv:1805.04803*, 2018.

[65] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017.

[66] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep reinforcement learning for dialogue generation," *arXiv preprint arXiv:1606.01541*, 2016.

[67] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[68] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," 2016.

[69] M. Henderson, B. Thomson, and J. D. Williams, "The third dialog state tracking challenge," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pp. 324–329, IEEE, 2014.

[70] E. Nouri and E. Hosseini-Asl, "Toward scalable neural dialogue state tracking model," *arXiv preprint arXiv:1812.00899*, 2018.

[71] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Association for Computational Linguistics, 2014.

[72] K. Hashimoto, c. xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple nlp tasks," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1923–1933, Association for Computational Linguistics, 2017.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization.," *CoRR*, vol. abs/1412.6980, 2014.

[74] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A

simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[75] Z. M. Ziegler and A. M. Rush, "Latent normalizing flows for discrete sequences," *arXiv preprint arXiv:1901.10548*, 2019.