

Distribution Agreement

In presenting this dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this dissertation. I retain all ownership rights to the copyright of the dissertation. I also retain the right to use in future works (such as articles or books) all or part of this dissertation.

James Finch

Date

Leveraging Diverse Data Generation for Domain-Adaptable Dialogue State Tracking

By

James Finch
Doctor of Philosophy

Computer Science and Informatics

Jinho D. Choi, PhD
Advisor

Joyce C. Ho, PhD
Committee Member

Fei Liu, PhD
Committee Member

Yang Liu, PhD
Committee Member

Accepted:

Kimberly Jacob Arriola, PhD, MPH
Dean of the James T. Laney School of Graduate Studies

Date

Leveraging Diverse Data Generation for Domain-Adaptable Dialogue State Tracking

By

James Finch
B.S., Michigan State University, MI, 2018

Advisor: Jinho D. Choi, PhD

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2023

Abstract

Leveraging Diverse Data Generation for Domain-Adaptable Dialogue State Tracking By James Finch

This work investigates improving domain adaptability in Dialogue State Tracking (DST), a crucial task for integrating conversational AI to real-world software applications. DST produces structured state representations that track important information in dialogue, which can be used as an interface to external software components and for controlling dialogue model behavior. However, obtaining DST models that can robustly adapt to new application domains is an ongoing research challenge. The proposed work aims to improve the utility of DST by making the domain adaptation of DST models more effective and cost-efficient. To achieve this, a new task is proposed called Dialogue State Generation (DSG). The goal of DSG is to infer both the schema and values of dialogue state in unseen dialogue domains, and experimental results demonstrate the effectiveness of the presented DSG approach for tackling the challenge of domain generalizability. The DSG approach is then extended for Slot Schema Induction, which is shown to be the first practical method for discovering a consistent set of new slot types from unlabeled data. Finally, the novel DSG and Schema Induction approaches are leveraged to generate a synthetic DST dataset with silver dialogue state labels that covers 1,000 different domains, an order of magnitude more than any existing dataset. An evaluation of few- and zero-shot DST models trained on the domain-diverse synthetic data demonstrates a substantial positive impact on DST domain adaptation. These contributions improve the feasibility of integrating conversational AI in real-world applications, taking steps towards the global improvement of software applications' efficacy and ease of use.

Leveraging Diverse Data Generation for Domain-Adaptable Dialogue State Tracking

By

James Finch
B.S., Michigan State University, MI, 2018

Advisor: Jinho D. Choi, PhD

Acknowledgments

I am deeply grateful to my advisor, Professor Jinho D. Choi, for his unwavering support and guidance throughout my doctoral journey. I also extend my heartfelt thanks to my committee members, Professor Joyce C. Ho, Professor Fei Liu, and Dr. Yang Liu, for their invaluable feedback and insights. I appreciate the contributions of Boxin Zhao, who was a wonderful collaborator during the final stages of my work. Thank you to my parents for supporting me, and to my sister Jody Finch and her fiancé Isaac Mazur for sharing in the struggles of graduate school. Finally, a thank you to my wife Sarah, for her wonderful partnership in both life and research.

Contents

1	Introduction	1
1.1	Application Utility of Conversational AI	2
1.2	Software Integration of Conversational AI	4
1.3	Challenges of Dialogue State Tracking	5
1.4	Research Contributions	6
1.5	Organization	7
2	Background	9
2.1	Task-Oriented Dialogue	10
2.2	Data for Task-Oriented Dialogue	11
2.3	Dialogue State Tracking	13
2.4	Slot Schema Induction	17
3	Emora: A Socialbot Built from Custom State and Policy Rules	19
3.1	Application Setting	20
3.2	Motivations for a Structured-State-Based Socialbot	22
3.3	Emora-STDm: Controllable Human-Computer Dialogue Using Structured Dialogue State	23
3.3.1	NATEX: Natural Language Expression	23
3.3.2	Dialogue Management	26
3.4	Design and Deployment of Emora	28

3.4.1	Conversation Design	29
3.4.2	Results and Analysis	31
3.5	Experimental Evaluation of Emora	34
3.5.1	Chatbot Selection	35
3.5.2	Evaluation Methods	36
3.5.3	Evaluation Study	37
3.5.4	Chatbot Evaluation	39
3.6	Discussion	41
4	Dialogue State Generation	43
4.1	Dialogue State Generation (DSG)	44
4.2	Related Work	45
4.3	GPTPipe: DSG with Zeroshot Pipeline	47
4.3.1	Question-Answer Pair (QA) Generation	48
4.3.2	Slot-Value Translation	49
4.4	DSG5K: New Diverse DSG Dataset	49
4.4.1	Scenario Derivation	50
4.4.2	Information Composition	51
4.4.3	Dialogue Generation	51
4.5	End-to-End (E2E) DSG Model	52
4.6	Experiments	53
4.6.1	Datasets	53
4.6.2	Models	54
4.6.3	Human Evaluation: Metrics	54
4.6.4	Human Evaluation: Results	55
4.6.5	Automatic Evaluation: Metrics	56
4.6.6	Automatic Evaluation: Results	58
4.7	Discussion	59

5	Slot Induction	61
5.1	DSG-I: Inducing Slots from DSG Inferences	62
5.1.1	Encoding Slot Value Candidates	63
5.2	Experiments	65
5.2.1	Data	65
5.2.2	Evaluation Metrics	66
5.2.3	Pilot Evaluation	69
5.2.4	Models	70
5.2.5	Results	71
5.2.6	Analysis	72
5.3	Discussion	74
6	Domain-Adaptable Dialogue State Tracking	77
6.1	Related Work	78
6.2	Approach	80
6.3	Data	83
6.4	Models	83
6.5	Results	85
6.6	Error Analysis	88
6.7	Discussion	88
7	Conclusion	90
7.1	Limitations and Future Work	91
7.2	Impact	93
	Bibliography	94

List of Figures

1.1	A real conversation between ChatGPT and a human user (abridged for readability). Structured information below turns such as <code>book day</code> is handcrafted to illustrate the necessary information needed for integration in live hotel reservation software.	2
2.1	Example conversation from the MultiWOZ dataset. User turns in orange are labeled with a state consisting of slot-value pairs.	10
3.1	A dialogue graph using a state machine approach with NATEX to dialogue management.	25
3.2	The average daily and last seven day user ratings for the time periods surrounding the current-event-focused openings. Single day openings are indicated by a red dashed line and multi-day openings are denoted by the red highlighted section in each graph.	33
3.3	The daily average user rating and the average seven day user rating received during the Quarterfinals and Semifinals.	34
3.4	Proportions of turns expressing undesirable behaviors, with 95% Wilson score confidence intervals.	39
3.5	Proportions of turns expressing desirable behaviors, with 95% Wilson score confidence intervals.	39
3.6	Average Turn Likert ratings of the conversations, with 95% Student's t confidence intervals.	41

4.1	Two example inferences of GPTPipe, a GPT-based DSG pipeline that infers dialogue state information by first generating question-answer pairs for important information shared in a given dialogue turn (Section 4.3.1), then translates questions to slot names and answers to values to obtain the final dialogue state update (Section 4.3.2).	47
4.2	Examples of the three stages in the data generation pipeline used to create DSG5K.	50
4.3	Example of input and output formats for end-to-end DSG using a sequence-to-sequence approach.	52
5.1	DSG-I approach to slot schema induction. The final description generation step is not included in the approach, but is shown for illustrative purposes.	62
5.2	SBERT encoding.	64
5.3	RoBERTa encoding.	65
6.1	An example of an input token sequence format used for training. The dialogue context c_i (yellow) and slot name s_i (peach) is taken from the synthetic DSG5K data, and extended with an automatically generated slot description d_i (green) and value examples e_i (red). Few shots f_i (purple) are optionally used in the few-shot setting and collected automatically using the DSG-I slot induction approach presented in Chapter 5. The value example $v_i = 20$ acres is the silver output sequence for this training example.	81

List of Tables

3.1	Simulated interaction with Emora, modeled on interactions observed from real users. User inputs are denoted as $U\#$. System responses involving multiple state transitions are denoted on separated contiguous lines as $E\#[a-b]$. The third column indicates the slot values tracked from a given user utterance. The fourth column displays the contents of the stack at each point in the conversation (CoS: covid_sympathy, CoE: covid_end).	30
3.2	Results of A/B test for fact-based and opinion-based coronavirus openings. (*) denotes a statistically significant result, $p < 0.10$	32
3.3	The average rating of conversations stratified by the types of components they contain. (*) denotes a statistically significant result, $p < 0.10$	32
3.4	The pilot results for 6 bots, showing the theme of the approach (Theme), the number of collected conversations (N), and the avg. dialogue-level Likert quality rating (Q). Pass denotes which models passed the verification criteria and were included in the full study.	36
3.5	The 8 labels for Likert evaluations, henceforth referred to using their abbreviations and colors.	37

3.6	The 16 behavior labels within ABC-Eval. Row separators denote evaluation task groupings. Bold indicates behavior labels kept in final set. [1] Gopalakrishnan et al. [17], [2] Higashinaka et al. [23], [3] Mehri and Eskenazi [42], [4] Mehri and Eskenazi [43], [5] Phy et al. [53], [6] Sanguinetti et al. [67], [7] Beattie et al. [3], [8] Sun et al. [73], [9] Xu et al. [87], [10] Rashkin et al. [61], [11] Smith et al. [71], [12] Majumder et al. [40], [13] Rashkin et al. [60], [14] Zhong et al. [95], [15] Zhou et al. [96], [16] Zhou et al. [97], [17] Gupta et al. [18], [18] Honovich et al. [24], [19] Santhanam et al. [68], [20] Shuster et al. [70], [21] Li et al. [37], [22] Nie et al. [49], [23] Welleck et al. [80], [24] Xu et al. [88].	38
4.1	The statistics of the DSG5K dataset (Section 4.4) with dialogue state update labels generated by GPTPipe (Section 4.3). SN/SV: slot names/values respectively, * <i>D/T/S/SN/SV</i> : * per dialogue/turn/scenario/SN/SV, respectively.	52
4.2	The statistics of our data split. TRN/DEV/TST: training/development/evaluation sets, DO: domains, DI: dialogues, TU: turns, SL: slots, US: unique slot names. . . .	53
4.3	Human evaluation of SGD-DSG, E2E-DSG, and GPTPipe on DSG5K and SGD for the completeness and correctness. * implies that the E2E-DSG result is statistically significant compared to the GPTPipe result.	56
4.4	Automatic evaluation results on the SGD evaluation set (Section 4.6.5).	58
4.5	Krippendorff’s Alpha scores for inter-annotator agreement (Human), or agreement of binarized automatic metrics with majority vote human judgments.	58
5.1	Slots present in each of the main 5 MultiWOZ domains.	66
5.2	Pilot Results	70
5.3	Example of 10 slot value inferences from the DSG model, sampled from a single cluster produced by DSG-I using the SBERT-S encoding strategy.	70

5.4	Schema induction results on the MultiWOZ 2.1. DA: Domain-aware evaluation setting. DF: Domain-free evaluation setting. *DSI and USI results are taken from Yu et al. [92].	71
5.5	Examples of DSG-I results mapping to the gold <code>Hotel</code> slots of MultiWOZ. Dashed lines separate individual DSG-I clusters that map to the same gold slot.	73
5.6	Error analysis	76
6.1	DST results on MultiWOZ, sorted within each division by avg. Joint Goal Accuracy (Eq. 2.1). Improvement to performance resulting from leveraging the presented data generation approach for pretraining provided in parentheses. * denotes the conjectured parameter size of OpenAI Codex. † denotes use of QLoRA.	85
6.2	Error analysis of DSG5K Zero-shot pretrained model Llama-PT on MultiWOZ . . .	87

List of Algorithms

1	Scenario Derivation	51
---	-------------------------------	----

Chapter 1

Introduction

Conversational Artificial Intelligence (AI) has recently gained much popularity, as it allows people to interface with technology in an intuitive way via natural language. Spoken conversational assistants such as Alexa, Siri, and Google Assistant have existed for several years now and allow users to perform simple tasks such as checking the weather or setting an alarm with speech input alone. More recently, instruction-tuned large language models (LLMs) such as ChatGPT have shown groundbreaking flexibility for interpreting user instructions to generate complex text output including emails, step-by-step tutorials, and even computer programs. There is an enormous potential for leveraging these current conversational AIs for the next generation of conversational assistants, as their broad natural language understanding and problem solving abilities should provide several advantages over traditional keyboard-and-mouse user interfaces. However, integrating conversational AI in real-world software systems requires tracking structured information introduced throughout conversations with users in order to interface with application software components. Tracking this structured information is known as Dialogue State Tracking (DST), and it is an open research task with many ongoing challenges. In particular, adapting DST models to the specific requirements of a given application setting in a cost-effective way and without a large performance degradation is difficult. This dissertation describes a novel body of research that tackles the problem of efficient domain adaptation for DST, facilitating the use of conversational AI in real-world applications.

1.1 Application Utility of Conversational AI

There are several advantages to integrating conversational AI into software applications. The conversation between ChatGPT and a human user in Figure 1.1 illustrates this potential using hotel booking as an example setting. In this dialogue, ChatGPT plays the role of a hotel receptionist talking to a hotel guest in order to make a reservation. Although this conversation was produced independently of any actual hotel booking software, one can see several advantages to the use of conversational AI if the conversation was indeed connected to a real-world hotel booking system.

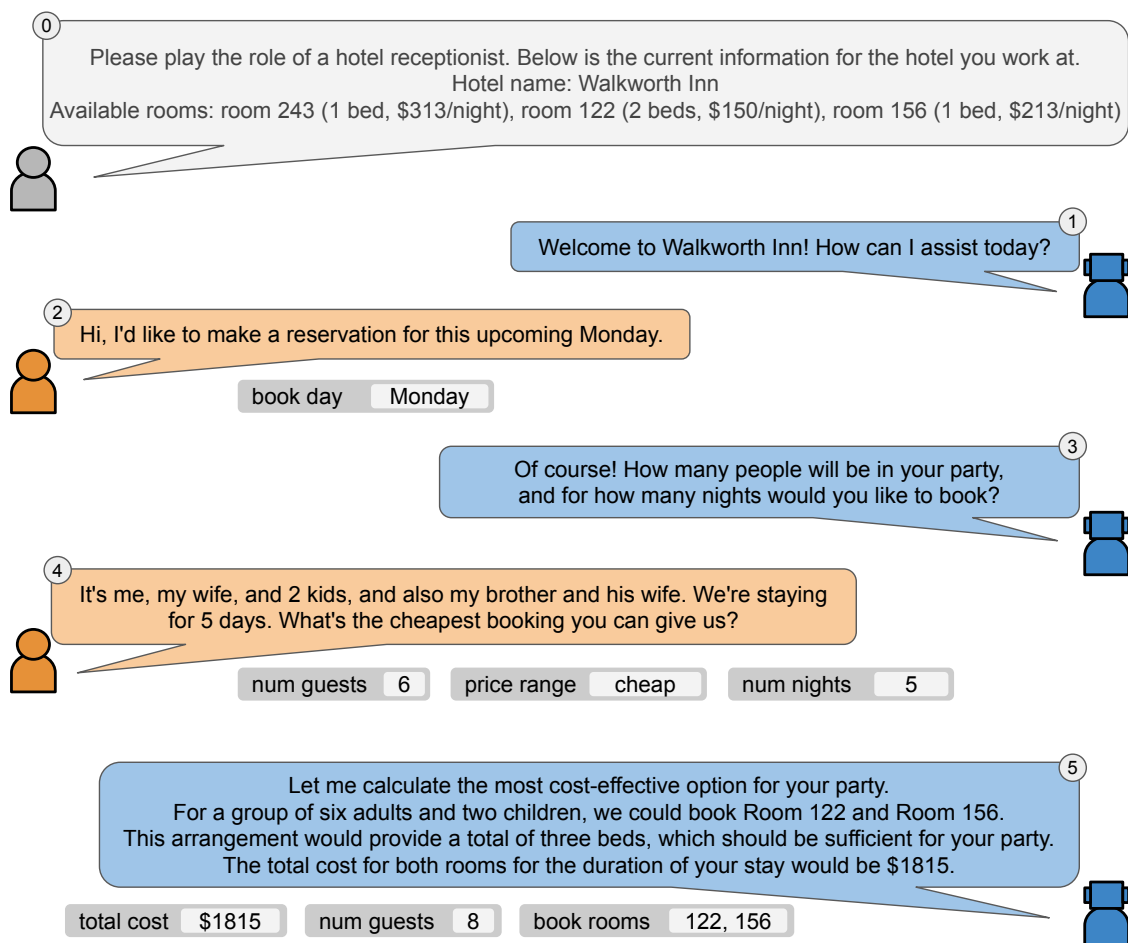


Figure 1.1: A real conversation between ChatGPT and a human user (abridged for readability). Structured information below turns such as `book day` is handcrafted to illustrate the necessary information needed for integration in live hotel reservation software.

Ease of Use While traditional keyboard-and-mouse interfaces require time and effort for users to become familiarized with the layout, menus, and capabilities of the interface, users can intuitively interact with conversational AI simply by speaking or typing in natural language even with no expertise. Furthermore, any domain knowledge the user lacks can be tutorialized by the AI as needed.

Adaptation to Underspecified Objectives When the user's objective is vague or underspecified, conversational AI can adapt gracefully by asking the user for additional details, or by inferring or recommending a concrete action plan that is likely to satisfy the user's needs. In turn 2 of the conversation in Figure 1.1, the user asks to make a reservation for Monday but does not specify the duration of the stay or which rooms to book, both of which are required for a reservation to be successfully registered. ChatGPT is able to adapt to this underspecification by asking for the number of guests and length of stay in order to make progress. Once the number of guests are provided in turn 4, ChatGPT makes a recommendation for which individual rooms to book, resulting in a complete action plan with all required information.

Flexible Problem Solving Conversational AI is also capable of solving problems in a way that is customized to a user's particular situation. In turn 5 of the conversation in Figure 1.1, ChatGPT correctly suggests the cheapest subset of available rooms that provide enough beds for the user's party. This kind of problem solving often relies on commonsense reasoning, such as who in the party is comfortable sharing a bed and how a party divides into separate rooms. Because it is difficult to anticipate the wide variety of factors that affect this kind of problem, such as the interpersonal relationships of the party, it is unlikely that any traditional booking software could provide this solution. Integrating conversational AI into software applications can therefore offload intellectual work typically performed by a human user when the software itself does not provide any programmatic solver.

1.2 Software Integration of Conversational AI

In order to integrate conversational AI with a particular software application, certain key information in the conversation must be translated into a structured representation based on the requirements of other software components. By obtaining this structured information from the conversation, the conversational AI can interface with other software application components in two necessary ways: Application Program Interface (API) calls and custom policies.

API Calls No matter how capable a dialogue system is at generating natural language responses, the system must be compatible with the APIs of external software components in order to be able to perform most real-world tasks. In the case of the conversation in Figure 1.1, although the conversation with ChatGPT is real, structured information from the conversation would need to be automatically extracted and sent through an API call to register the final reservation in a live hotel booking application. For example, the hypothetical application's reservation API might require a list of room numbers, the day of the upcoming week the stay begins, and the duration of the stay in order to register a reservation in a database. This information could be represented by the information slots `book rooms`, `book day`, and `num nights` respectively, as shown in Figure 1.1. Similarly, in order to access information about available rooms like those provided in the turn 0 prompt, the reservation system might require `book day` and `num nights` to perform a relevant database query, then send the query results back to the AI as text input to provide real and up-to-date information for continuing the conversation.

Custom Policies Real-world applications frequently carry requirements that originate not from the user or application structure, but rather from a third party that has a stake in application performance. For example, suppose a hotel manager wants the conversational AI integrated with their booking software to specially mention that the hotel provides free parking and free wifi if the user indicates a cheap price range. If the user's `price range` is explicitly tracked as structured information during their conversation with the AI as in Figure 1.1, the dialogue system

would be able to inspect the user price range to dynamically control the AI's behavior, such as by switching the prompt at turn 0 with the instruction: "Please inform the guest that Walkworth Inn provides free wifi and free parking.". Tracking structured information also makes it possible to control fine-grained aspects of the AI's response. Turn 5 in Figure 1.1 contains a mistake where ChatGPT miscounts the number of guests, but by tracking `num guests` as structured information and checking for discrepancies, the response can be automatically revised before it is sent to the user.

1.3 Challenges of Dialogue State Tracking

Despite their powerful natural language understanding and text generation abilities, current conversational AI models like ChatGPT fall far short of their potential to be integrated into software applications due to challenges in extracting structured information for use in API calls and custom policies. The task of inferring structured application-relevant information throughout a dialogue is known as Dialogue State Tracking (DST), and it is an open research area. The major challenges of DST are outlined below.

Domain Adaptation Each software application has different requirements for the structure of information needed to integrate conversational AI, resulting in the need for DST to adapt to the information type semantics of any particular application domain. It is challenging to perform this domain adaptation without incurring a large cost for data collection in the target domain. Often the performance of a general DST model will degrade substantially when adapted to unseen target domains in a data-efficient way, reflecting the realistic usage of DST models in practice.

Cost of Collecting Adaptation Resources When adapting a DST model to a particular application domain, a set of information types (slots) for the model to track is necessary but often requires a time-consuming development process to define the slots by hand. Furthermore, for effective domain adaptation to be possible, natural language descriptions or example data for each slot must

be collected or handcrafted. These development and collection costs are a barrier for integrating conversational AI in real-world applications, and slow down new feature development for existing conversational AI applications.

Efficiency In practical applications, there may be hundreds of slots that must be tracked by a DST model to support integrating conversational AI. This can easily create a latency bottleneck in a conversational AI system if a computationally intensive DST model is used such as a LLM. This latency bottleneck can be especially detrimental because users expect timely responses when conversing with the AI. Thus, finding efficient methods for tracking structured information remains a significant challenge in the integration of conversational AI into software applications.

1.4 Research Contributions

The efficient domain adaptation challenge of DST is the focus of the research contributions presented in this dissertation, in which the following research questions are tackled.

Is dialogue state useful for open-domain and chat-oriented dialogue systems? Related work focuses on the use of DST in task-oriented dialogue systems, where the goal is to accomplish a concrete task through dialogue, and tracked dialogue state is used primarily as a means of filling API call parameters. However, the presented work demonstrates that structured dialogue state is also a powerful tool for enabling rich open-domain, chat-oriented conversation where there is no concrete conversation objective other than engaging the human user. This finding motivates the need for domain-general dialogue state inference, and exemplifies the rich controllability of dialogue system behavior that is afforded by structured dialogue state information.

Can domain-general dialogue state inference be achieved? Conventional DST requires adapting a DST model to a particular set of application slots to be used in practice. This adaptation to the target domain is difficult, especially when the target domain is unlike any data used to train the

base DST model. The presented work tackles the domain generalizability challenge by formulating a new task called Dialogue State Generation (DSG), in which a model jointly discovers the state schema and infers state values in a target domain without any domain adaptation step. Although the presented DSG model does not produce consistent state schemata within the same domain, it successfully achieves domain generality and serves as a potent resource for increasing the diversity of DST training resources.

How can dialogue state schemata be automatically induced from unlabeled dialogues? Adapting DST models to new domains conventionally incurs a steep cost of human labor, as the state schema of the target domain must be manually specified. In the event of unlabeled dialogue data being available for the target domain, an automatic solution for inducing the state schema would thus drastically reduce the cost of DST domain adaptation. The presented schema induction approach leverages the presented DSG model to induce a unified schema across unlabeled dialogue examples, and is shown to be the first practical approach for this task.

Can DST domain adaptability be improved by increasing training data diversity? Existing DST training resources are quite limited in their coverage of different task domains and types of information tracked, which is hypothesized to limit the performance of DST models after adapting to unseen target domains. This hypothesis is tested in the presented work by training DST models on a new dataset covering over 1,000 domains, which is made possible by automatically annotating dialogues using a DSG model. Experiments demonstrate a large performance improvement on unseen target domains for DST models that make use of diverse training resources.

1.5 Organization

Chapter 2 provides background on Dialogue State Tracking (DST) and related tasks, highlighting the major challenges in this area. Chapter 3 provides a case study of the deployed open-domain dialogue system Emora, which demonstrates the utility of tracking structured dialogue state in

an understudied setting. Chapter 4 presents the novel Dialogue State Generation (DSG) task formulation and model that achieves domain-general dialogue state inference. Chapter 5 details a simple yet effective approach for leveraging the DSG model to induce dialogue state schemata from unlabeled data. Chapter 6 presents experiments on DST that show the large impact on domain adaptability provided by diverse training resources. Finally, Chapter 7 outlines the key takeaways, limitations, and future directions of the presented research.

Chapter 2

Background

This chapter describes the current state of the art in Dialogue State Tracking (DST) and related work, and situates DST within the context of human-computer dialogue research. Broadly, human-computer dialogue research can be divided into Task-Oriented and Chat-Oriented work, with DST traditionally serving as a foundational subtask of Task-Oriented Dialogue.

Task-Oriented Dialogue research focuses on developing human-computer dialogue systems for completing well-defined tasks through natural language, such as booking flights, making restaurant reservations, or setting an alarm [5, 62]. In these dialogues, the human generally plays the role of a user who brings certain custom requirements to the task scenario, and the automatic dialogue system must meet their request by providing information, requesting user preferences, and taking actions to accomplish the task. Task-oriented dialogue is generally straightforward to evaluate, as the task success rate of a dialogue system’s conversations is directly measurable.

Chat-Oriented Dialogue research focuses on developing human-computer dialogue systems that are able to support open-ended conversations with humans. Although a theme or topic might be chosen to give some guidance to the conversation scenario, such as conversations about Wikipedia articles [11], conversers are generally free to ask about or share any information, opinions, or experiences they find interesting [64]. For this reason, most chat-oriented dialogue tasks are

considered *open-domain*, as opposed to task-oriented dialogues that are generally closed to the task domain. As there is no concrete task objective, chat-oriented dialogue evaluation relies on human judgements on the quality of model responses, making evaluation challenging from the labor cost and variance in human judgements due to subjectivity [15].

2.1 Task-Oriented Dialogue

Task-oriented dialogue systems are traditionally modularized into 3 components, each of which is a subtask with a dedicated body of research: Dialogue State Tracking (DST), Policy Learning (POL), and Response Generation.

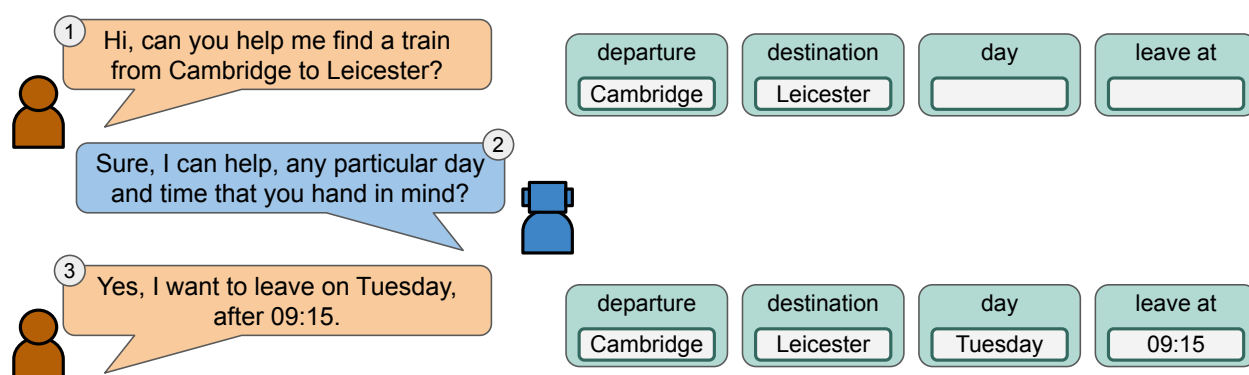


Figure 2.1: Example conversation from the MultiWOZ dataset. User turns in orange are labeled with a state consisting of slot-value pairs.

Dialogue State Tracking aims to maintain a structured representation of all information in the dialogue required to effectively act and respond towards completing the task. Although in theory any structured state representation is possible, the dialogue state is conventionally represented as a set of n slot-value pairs $\{(s_1, v_1), (s_2, v_2), \dots, (s_n, v_n)\}$, where the slots s_1, s_2, \dots, s_n represent the *slot schema*, a predefined set of information types associated with the task. Seen in Figure 2.1, all slots are initially empty at the start of the conversation by default, and are filled in with values tracking the current state as the conversation progresses. Although system actions and knowledge are encapsulated in the dialogue state in practice, these are trivial to track since they originate from

the system itself; thus DST research focuses on the problem of tracking state information originating from the user as shown in Figure 2.1. Further details regarding recent work in DST are given in Section 2.3.

Policy Learning aims to automatically learn a function $P : \mathcal{S} \rightarrow \mathcal{A}$ mapping state space \mathcal{S} to action space \mathcal{A} . This function P determines the action to be taken in each state. For example, given the tracked dialogue state shown in turn 3 in Figure 2.1, the dialogue system policy might produce the action `inform(choice=2)`, which reflects a decision to inform the user that there are two trains matching the user’s search criteria. Note that it is possible and common in practice to create or augment P using handcrafted rules that check the slot state for conditions to trigger certain actions. However, since handcrafting the policy can be labor intensive and produce suboptimal policies, reinforcement learning methods have been explored to learn the policy automatically [28, 32]. Recently however, there has been a trend to learn a policy jointly with the response generation model using a single end-to-end neural approach [52, 8].

Response Generation aims to produce a natural language response given the current dialogue state. This is possible using handcrafted response templates, as the tracked state should provide all necessary information to produce coherent and productive responses; however, this component is generally handled by a neural response generation model as in Peng et al. [52], Chen et al. [8].

2.2 Data for Task-Oriented Dialogue

Many datasets have been created for Task-Oriented Dialogue, [55, 5, 62] but most are substantially limited in size, domain diversity, and naturalness [55, 62] due to the steep costs of data collection. Two datasets, MultiWOZ [5] and Schema Guided Dialogues (SGD) [62], are by far the most popular due to their relatively high volume of dialogue examples and domains.

MultiWOZ The MultiWOZ dataset [5] was collected using a Wizard-of-Oz (WOz) set-up in which two crowd workers use a custom software application to have a natural language dialogue. Each worker is provided a role, either system or user, and given instructions describing their role and goal for the conversation. The conversation goals given to workers were based on 5 task domains inspired by the situation of a user talking to an automated travel agent. For example, in the "hotel" domain, the user might be given the goal to book an expensive hotel on the east side of the city. The details of task goals correspond to a set of predefined slots within each task domain in order to support dialogue states that incrementally update over the course of a dialogue as more goal information is specified between the two conversers. To obtain state labels, another set of crowd workers annotated each turn of the "user" role. The final dataset contains 8,438 dialogues spanning 5 domains¹ with 31 total slot types².

Schema Guided Dialogues The Schema Guided Dialogues (SGD) dataset [62] was collected using an automatic approach to generate synthetic dialogues from predefined domain schema definitions, which were then revised by crowd workers to improve the naturalness and coherence of each dialogue turn. A set of tasks corresponding to domains was first defined with slots and a list of possible values for each slot. Then, a simulated user and system agent modelled using probabilistic automata are initialized with conversation goals and take turns updating the dialogue state using a set of transition rules. The result is a sequence of conversation states that can then be translated into a dialogue using a set of handcrafted templates, and then further paraphrased by human crowdworkers into reasonably natural conversations. SGD focuses on tasks representative of a user talking with a virtual assistant such as Alexa or Siri, and therefore includes domains such as "alarm", "weather", and "flights". The dataset includes 16,142 dialogues spanning 16 domains and 214 slot types. SGD is therefore the most domain-diverse and largest DST dataset, although its dialogues are arguably less natural than those in MultiWOZ.

¹The Bus and Hospital domains are excluded by convention due to a poverty of examples.

²There are 17 slot types in MultiWOZ if slots from different domains that represent the same kind of information are counted as a single type

2.3 Dialogue State Tracking

The task of dialogue state tracking (DST) is to produce a structured dialogue state representation for each turn in a dialogue, where the dialogue state is represented as a set of slot-value pairs. Slots are information types defined by a domain-specific schema such as "destination" or "arrival time", and values represent corresponding information that has been shared in the dialogue such as "New York" or "5:30". Formally, given a dialogue context $D_{1..i}$ from turn 1 to some turn i , and a task-specific slot schema defining a set of n slots T , a dialogue state tracker must predict a dialogue state $S_{D_i} = \{(s_1, v_1), (s_2, v_2), \dots, (s_n, v_n)\}$ representing all slot information in every turn of the dialogue context. A special value n/a can be used to represent empty slots. The performance of DST models is conventionally measured using Joint Goal Accuracy (JGA), which is the proportion of turns in a test set where *all* slot-values in the predicted state exactly match all slot-values in the gold state:

$$JGA = \frac{\sum_{i=1}^N \mathbb{I}(\hat{S}_{D_i} = S_{D_i})}{N} \quad (2.1)$$

Traditionally, DST has been tackled using supervised learning, where the model learns to predict values for each slot by fitting to a large number of gold examples in the training dataset [83, 90, 86]. However, this approach is usually impractical in application settings where such training data is unavailable. Therefore, several lower-resource settings have been proposed to test the ability of an approach to correctly adapt to the provided slot schema. These include the low-resource setting, few-shot setting, and zero-shot setting. Each of these lower-resource settings limit the number of in-domain gold-labelled examples, testing whether a DST approach is capable of adapting to new target domains and slot types. For evaluation in these settings, a leave-one-out methodology is commonly used, where k independent evaluations are performed for each of k domains in a dataset. In each evaluation, the full set of gold-labelled training data is observable for the $k - 1$ training domains, but only a limited number of training examples are observable on the held-out target domain to simulate a low-resource scenario. The low-resource and few-shot settings respectively

provide a small or very small number of training examples in the target domain during test time, and the zero-shot setting provides no in-domain training examples. All settings leverage end-to-end neural models trained to tag or generate slot values, with pretrained transformer-based [76] models used most commonly in recent work to good effect [30, 19, 94, 22, 83].

Low Resource Dialogue State Tracking In this setting³, a small number of gold, in-domain training examples are provided for each slot type in the task schema. Common downsampling percentages are 1, 5, and 10%; however, since downsampling even of 1% of the training data usually provides hundreds of gold-labelled examples to be used for training, this setting still reflects a costly approach that requires substantial data collection effort.

The current state-of-the-art (SoTA) approaches in low-resource DST involve two steps: (1) retrieving one or more similar examples from the small number of in-domain observable examples to use in a natural language prompt, and (2) generating a value prediction using a sequence-to-sequence (S2S) model that takes both the prompt and dialogue context as inputs [26, 30, 7]. Two of these works use OpenAI Codex as their generation model [26, 30], and outperform the best non-LLM implementation that follows the same approach [7].

Few-Shot Dialogue State Tracking In the few-shot setting, a very small (usually 1 to 10) number of in-domain gold-labelled examples ("shots") are provided for each slot type. Each example takes the form of a single sample one would typically see in a fully supervised setting where a dialogue context is paired with a gold slot-value label. In this setting, the definition of each slot is specified using the example shots, but the quantity of examples is too small for typical supervised approaches to be effective. Therefore, approaches in this setting often treat the few shots as additional input when predicting each slot value, which trains the model to condition each value prediction on the provided examples. A model that learns to condition its prediction on the provided examples in a sufficiently general way will then be capable of correctly predicting values for new unseen slots

³The majority of work in low-resource DST refers to the task setting as "few-shot" instead of "low-resource". I break the conventions of this terminology in order to keep in line with other work in few-shot NLP and distinguish this work from task formulations where truly only a few (1-10) shots are available.

during test time as long as a sufficient number of examples for those unseen slots are also provided as input. Unlike fully supervised and low-resource setting, the few-shot dialogue state tracking setting is a good proxy for application scenarios. Dialogue system developers are usually able to handcraft or collect a few examples per slot type, whereas it is often prohibitively time consuming or costly to collect enough examples for supervised training even in the low-resource setting.

Unfortunately, the few-shot setting in DST is underexplored and there is poor standardization for evaluation methods, leading to difficulties when comparing approaches. Wu et al. [84] uses a sequence-to-sequence (S2S) approach that linearizes the dialogue context, slot type shots, and a question specifying the slot type, all into a single prompt to predict the slot value. They evaluate 5- and 10-shot settings on the unpopular MixSNIPS/MixATIS [55] dataset. Aksu et al. [1] take advantage of gold state labels to create response templates and a state transition graph from in-domain shots, and then generate entirely new dialogues by traversing several paths of this transition graph. Once a new state transition path for a synthetic dialogue is sampled from the transition graph, the turns from the original dialogues corresponding to each transition are used as templates and filled with new slot values to produce a final natural language dialogue. Synthetic dialogues are then used to train a DST model for the target domain. This approach demonstrates the ability for data generation to compensate for resource-poor situations, although their reported JGA on MultiWOZ 2.1 is substantially lower than SoTA zero-shot approaches on the same data. Gupta et al. [19] achieve strong performance on MultiWOZ 2.1 and Schema Guided Dialogues using a simple 1-shot S2S approach that predicts values given the dialogue history and a single shot for each target slot type. Since they limit to 1-shot, their approach can be almost-fairly compared with zero-shot approaches, and is competitive with even SoTA LLM zero-shot approaches at 65.9% JGA.

Zero-Shot Dialogue State Tracking The zero-shot setting is similar to the few-shot setting, except a natural language description of each slot is provided as the means of encoding the slot definition, rather than a small number of examples. Thus, approaches to few-shot and zero-shot dialogue state tracking are often compatible with the slot definition (which can be either the few-

shot examples or description) being provided as additional input to the model when predicting slot values. This setting is more challenging than the few-shot setting, however, because the provided slot definitions do not resemble the dialogue context from which the model is predicting values; instead, the model must interpret the natural language slot definition in order to correctly predict the appropriate value. On the other hand, this setting is even more cost-efficient than the few-shot setting since it usually requires far less human labor to specify slot descriptions than to handcraft or collect several few-shot examples.

The best approaches to zero-shot DST leverage slot descriptions use simple sequence-to-sequence (S2S) formulations to predict a value appropriate for the given slot description [94, 75, 78]. This S2S modelling formulation has been shown to be a simple but effective tool for adapting to new slot types, since models can leverage the given description of a new, unseen slot type when making predictions. Tavares et al. [75] use a natural language question in place of a traditional slot description to formulate zero-shot DST as a kind of question answering, showing that the semantics and framing of the description can significantly affect task performance. Wang et al. [78] extend the basic description-driven S2S approach with a mixture-of-experts architecture, where each expert is a simple S2S model focusing on a different semantic subspace for the task. The description-driven S2S approach for zero-shot DST has recently been explored using instruction-tuned large language models (LLMs), for example Heck et al. [22] demonstrates that ChatGPT is capable of outperforming traditional zero-shot DST approaches with a simple prompt. The current SoTA zero-shot model⁴ is by Gupta et al. [19], who use a T5-11B model in a simple S2S formulation to achieve 64.4% JGA on MultiWOZ 2.1, showing the power of fine-tuning models with larger parameter sizes for this task.

⁴Excluding few-shot models that use examples as "descriptions."

2.4 Slot Schema Induction

One of the challenges of applying DST models in practice is specifying a slot schema that defines which slots the DST model should track. In applications where a dialogue model is being integrated into existing software framework, the slot schema is typically specified by hand to match the existing infrastructure such as API call parameters or database schemata. However, in many cases, the external software components interfacing with the dialogue system are under development, and the choice of slots to track is made in tandem with the development of new features. In these cases, a typical developer workflow for creating the slot schema involves manually reviewing unlabeled dialogue data produced by a previous version of the dialogue system or by human-human interaction, and deciding which information is important to track as slots. This workflow is especially prevalent when implementing custom policy changes to the dialogue system behavior, where a developer seeks to control system behavior via programmatic intervention as a response to viewing a poor-quality dialogue produced by their system.

Slot schema induction (SSI) aims to reduce the cost of manual schema creation by automatically inducing slots from unlabeled dialogue data that are likely to facilitate task completion. The primary challenges of this task are (1) to correctly identify what information in the unlabeled dialogues would be important to track as slot values, and (2) aggregating candidate value information into a consolidated set of useful slot types. As SSI is a relatively recent task with few works, there is yet no standardized task formulation and evaluation method. However, several of the most recent works present similar variants of the following formulation.

Given a list of dialogues D , produce a list of slots $\hat{S} = [\hat{s}_1, \hat{s}_2, \dots, \hat{s}_n]$ where each slot \hat{s}_i is a cluster of values $\hat{s}_i = [\hat{v}_1, \hat{v}_2, \dots, \hat{v}_{|\hat{s}_i|}]$ corresponding to a gold reference slot $s_i = [v_1, v_2, \dots, v_{|s_i|}]$. To evaluate, a mapping $M : \hat{S} \rightarrow S \oplus [\text{none}]$ is constructed by hand or using a semantic similarity estimator to assign each predicted cluster to a gold slot (or to none if no gold slot matches). In this formulation, producing a name for each induced slot is out of scope; the goal is only to find clusters of values that correspond well to gold slots. Given the mapping M from predicted to gold slots, precision, recall, and F1 are calculated to measure the quality of induced slot clusters.

Approaches to slot schema induction follow a two-step pipeline of first extracting value candidates from dialogue text, then clustering dense semantic vector representations of each candidate to produce slot clusters. Hudeček et al., Wu et al., Qiu et al. [27, 85, 56] leverage tagging models trained on other tasks such as Semantic Role Labeling (SRL) or Named Entity Recognition (NER) to identify a set of value candidates from spans in the unlabelled dialogue data. Then, each candidate value is embedded into a dense vector representation using an encoder model such as BERT [10]. Finally, a clustering algorithm is used to induce the slot types from the value candidate span representations, often in conjunction with a filtering step to eliminate noisy candidate values. Hudeček et al. [27] use a simple cosine-similarity-based threshold to filter value candidates and merge them into clusters. Wu et al. [85] use an iterative, self-supervised method to alternate between fine-tuning the candidate value tagging model and cluster candidates, where the tagger learns to select candidates near the centroid of induced clusters. Instead of using NER and SRL models to identify value spans, [56] train a domain-general value tagger to identify value candidates, then use out-of-the-box clustering algorithms such as K-Means⁵ to induce slot types. One fully unsupervised approach for slot schema induction by Yu et al. [92] uses a PLM attention distributions between tokens in a dialogue turn to infer the syntactic structure of the turn, and treat constituents as candidate values. Three clustering steps are then followed to obtain a cluster hierarchy, where different encodings of the span candidates are clustered at each step using HDBSCAN [41].

Overall, the SoTA in slot schema induction is encouraging but not practical. The reported recall of predicting slot clusters that match gold slot clusters is quite high with the SoTA achieving 94% recall on MultiWOZ. However, most slot induction methods induce a number of slots that is about 10x the number of gold slots in a given domain, which still requires a large human effort to manually name and consolidate the clusters into a usable slot schema. Furthermore, a limitation to all previous work in this area is that value candidates are always extracted spans from the original dialogue text. This limits the ability of these approaches to induce abstractive slots, whose values rarely appear in the dialogue directly but are nevertheless critical to tracking the dialogue state.

⁵Incurrs the limitation of requiring the number of induced slot clusters K to be specified ahead of time.

Chapter 3

Emora: A Socialbot Built from Custom State and Policy Rules

Traditionally, research in dialogue systems has emphasized the use of dialogue state tracking (DST) in task-oriented settings, where the primary objective is to fulfill specific tasks through dialogue, typically by populating API call parameters. However, this chapter highlights the versatility of structured dialogue state in facilitating open-domain conversation in a chat-oriented setting, where the goal is simply to engage the human user in an interesting conversation on a variety of topics. Dialogue state is shown to be useful for open-domain and chat-oriented dialogue specifically because of the flexibility it affords dialogue systems developers in implementing custom policies to control dialogue system behavior. However, to achieve these advantages, fine-grained slots must be tracked that are extremely situational and application-specific. This is demonstrated by way of a case study of the chat-oriented dialogue system named Emora, which was deployed to a broad user base in a live application setting. An analysis of Emora's abilities resulting from its DST-driven design is provided by evaluation metrics collected from both the live deployment setting and experimental setting where it is compared to dialogue models from contemporary SoTA research.

3.1 Application Setting

Emora was deployed to live users on Amazon Alexa devices for a period of about 12 months as a part of the 3rd Amazon Alexa Prize Competition [16] and associated Alexa Skill. The main aim of the competition is to develop conversational agents capable of engaging in coherent and meaningful discussions with humans for as long as possible, while receiving high user feedback ratings. 10 participating universities built socialbots that interact with real users through Alexa in real-time, multi-turn, spoken dialogues.

To interact with the application, users initiate conversations with Alexa by saying phrases like "Alexa, Let's Chat." They are then connected to a randomly-selected one of the 10 socialbots participating in the competition, one of which is Emora. After ending the conversation, users are asked to verbally rate their experience, followed by an opportunity to give additional feedback. Conversation transcriptions with their associated ratings, if given by the user at the end of the conversation, are provided to participating teams in order to support scientific experimentation and improvements to their socialbots. By the end of its deployment, on the order of 1,000,000 conversations had been produced between Emora and users.

Although the application setting is intended to be open-ended in spirit, there are several characteristics of the application setting that distinguish it from a typical academic experimental setting, which are outlined below.

Alexa Skill Platform The Alexa Prize skill is a single application hosted within Amazon's Alexa Skill platform which has thousands of other applications available. Because the Alexa devices use a speech-only interface which relies on noisy automatic speech recognition (ASR), speech input that the user intends as a command to exit the Alexa Prize skill must be distinguished from speech intended to continue a conversation as a part of each deployed socialbot's natural language understanding capabilities. Furthermore, it is quite common for users to connect to the Alexa Prize skill and enter a conversation with a sociabot by accident, due to false positive classification of the invocation command "Alexa, Let's Chat". Detecting accidental connections and gracefully

exiting the conversation is therefore a desirable property of socialbot behavior to avoid poor user experiences.

Topic Restrictions In order to maintain the integrity of both the competition and Amazon Alexa as a product, certain sensitive conversation topics were required to be off-limits, with the expectation that socialbots would never introduce these topics and would decline interaction if the user introduced them. Some of these restricted topics included giving opinions on politics or religion, sexual topics, and hate speech.

Required Responses Some specifications were put in place for required responses the socialbots must give to certain questions. For example, if asked "who are you?" by a user, socialbots were required to distinguish themselves from Alexa as an Alexa Prize socialbot while maintaining anonymity to maintain competitive integrity.

Longitudinal Deployment Because Alexa Prize socialbots were deployed for up to a year, distributional shifts in user interests occurred over time in response to evolving current events. This was especially true for the duration of Emora's deployment, which coincided with the outbreak of the COVID-19 pandemic. Providing support for conversation in a way that matched the changing interests of the user base was ideal for user engagement. Additionally, current events such as holidays and recent news are one of the few sources of common ground among most users that can be leveraged as introductory topics to begin conversations in a reliably natural way.

Returning Users Although users were anonymous, some users would frequently invoke the Alexa Prize skill and converse multiple times with the same socialbots. In order to support the development of long-term, multi-session interactions with users, anonymous user identifiers were made available for each conversation, allowing data saved from previous conversations to be imported into future conversations with the same user.

3.2 Motivations for a Structured-State-Based Socialbot

When Emora was originally designed, the first approaches considered included end-to-end neural response generation and pipelines involving the use of several core natural language processing (NLP) models such as topic and intent classifiers, semantic parsers, and neural DST models. These approaches fell in line with contemporary research on human-computer dialogue. However, when prototyping various architectures, these approaches were quickly found to be insufficient for developing a viable socialbot for the Alexa Prize application setting for several reasons.

Imprecise Natural Language Understanding SoTA Natural language understanding (NLU) models produced by contemporary research tended to focus on fixed, generic schemata for NLU output. For example, named entity recognition work focuses on identifying mentions of entities, but the schema of entity types only covers generic classes such as `person` and `organization` [79]. This level of granularity for NER and other tasks provided insufficient information to effectively distinguish key dialogue states, such as whether the user had invoked the Alexa Prize skill on accident, was introducing a restricted topic, or personally knew someone infected by COVID-19.

Lack of Control Although it is possible to develop new NLU models using finer-grained or application-specific schemata, the cost barrier of collecting labeled data to train neural models is too high to be viable, especially for supporting conversation on current events. Few-shot approaches that allow adaptation of an NLU model to schema extensions such as few-shot NER [12] or zero-shot DST [19] might have been effective, but were understudied and suffering from poor performance at the time of Emora’s development. Similarly, neural response generation models supporting fine-grained control over response semantics were emerging research [74], meaning there was no reliable way of controlling response generation behavior for these approaches apart from collecting a large number of training examples [65].

In response to these challenges, an approach and implemented dialogue system development

framework, named Emora-STDM, were developed to support rapid development of human-computer dialogue systems using structured dialogue state and a handcrafted approach to both NLU and response generation. The next section describes Emora-STDM. The following Section 3.4 covers the design and deployment of Emora, a case study usage of this approach to dialogue.

3.3 Emora-STDM: Controllable Human-Computer Dialogue Using Structured Dialogue State

Emora State Transition Dialogue Manager, henceforth E-STDM, is an approach to human-computer dialogue affording rich controllability of dialogue system behavior by leveraging structured dialogue state representations. The goal of this approach is primarily to tackle the challenge of rapidly developing a human-computer dialogue system with full customization of system behavior. To achieve this, E-STDM does not rely on any data-driven approaches for NLU, development of the dialogue policy, or response generation, which typically require costly data collection in order to support custom system behaviors. To achieve this, E-STDM has three major components. NATEX is a new language for flexibly specifying grammars for parsing-based DST and response generation. The Dialogue State Machine controls the dialogue policy, selecting a candidate subset of NATEX NLU patterns and response generation templates at each turn. Finally, Information State Update Rules provide additional control for managing the dialogue state representation. E-STDM was implemented as a publicly available python package¹.

3.3.1 NATEX: Natural Language Expression

The challenge of understanding user input in natural language is addressed by NATural language EXpression, NATEX, a rule-based and fast-to-develop pattern matching system for intent classification and slot tracking. NATEX defines a comprehensive grammar to match patterns in user input by dynamically compiling to regular expressions. This dynamic compilation enables abstracting away

¹<https://pypi.org/project/emora-stdm/>

unnecessary verbosity of regular expression syntax, and provides a mechanism to embed function calls to arbitrary Python code. Critically, NATEX patterns are capable of filling slots by compiling regex variable captures. The key features of NATEX are highlighted below.

String Matching It offers an elegant syntax for string matching. The following NATEX matches user input such as *'I watched avengers'* or *'I saw Star Wars'* and fills the slot `$MOVIE` with the values *'Avengers'* and *'Star wars'*, respectively:

```
[I {watched, saw}
    $MOVIE={Avengers, Star Wars}]
```

The direct translation of this NATEX to a regular expression would be as follows:

```
.*?\bI\b
.*?(?:\b(?:watched)\b|\b(?:saw)\b)
.*?(?:P<MOVIE>(?:\b(?:avengers)\b|
                \b(?:star wars)\b)).*?
```

As shown, NATEX is much more succinct and interpretable than its counterpart regular expression.

Function Call It supports external function calls. The following NATEX makes a call to the function `#MDB` in Python that returns a set of movie titles:

```
[I {watched, saw} $MOVIE=#MDB()]
```

This function can be implemented in various ways (e.g., database querying, named entity recognition), and its NATEX call matches substrings in the user input to any element of the returned set. Note that not all elements are compiled into the resulting regular expression; only ones that are matched to the user input are compiled so the regular expression can be processed as efficiently as possible.

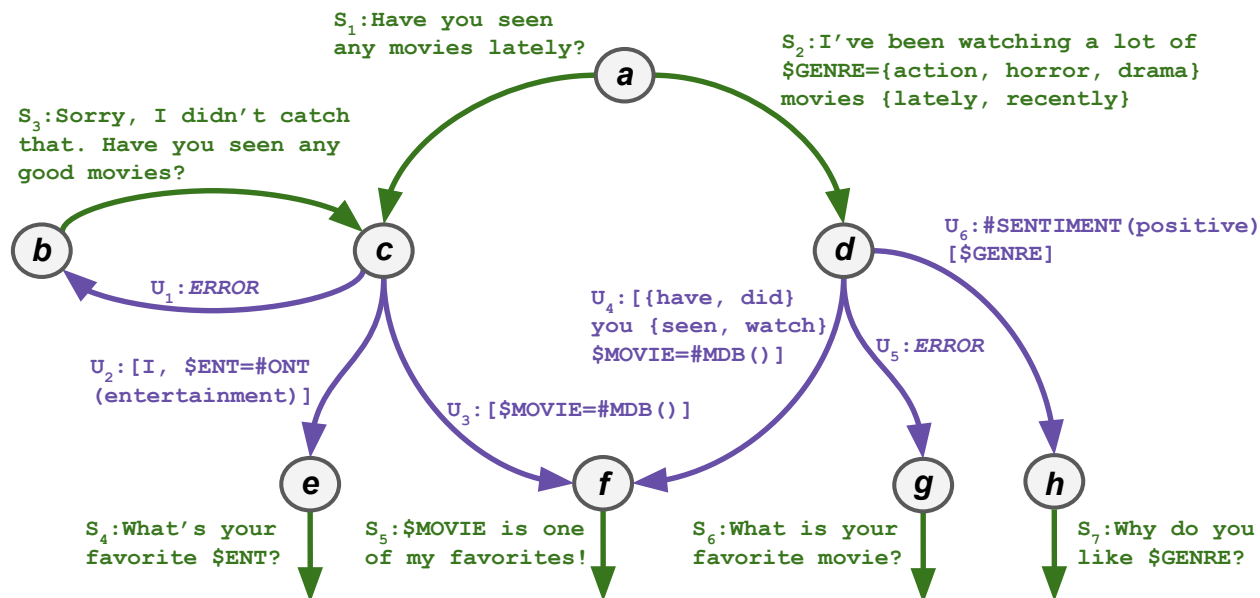


Figure 3.1: A dialogue graph using a state machine approach with NATEX to dialogue management.

Ontology It supports ontology editing and querying as the built-in NATEX function called #ONT. This ontology-based matching mechanism is a core mechanism for slot filling based on recognizing keyword lemmas that belong to a type set defined in the ontology. An ontology can be easily built and loaded in JSON. #ONT(movie) in the example below searches for the movie node in a customizable ontology represented by a directed acyclic graph and returns a set of movie titles from the subgraph of movie:

```
[I {watched, saw} $MOVIE=#ONT(movie)]
```

Response Generation It can be used to generate system responses by randomly selecting one production of each disjunction (surrounded by {}) in a top-down fashion. The following NATEX can generate “I watched lots of action movies lately” or “I watched lots of drama movies recently”, and assign the values of ‘action’ and ‘drama’ to the slot \$GENRE respectively:

```
I watched lots of $GENRE={action,
horror, drama} movies {recently, lately}
```


Error Checking The NATEX compiler uses the Lark parser to automatically detect syntax errors.² Additionally, several types of error checking are performed before runtime such as:

- Call to a non-existing function.
- Exceptions raised by any function.
- Function returning mismatched type.
- Reference to a non-existing variable.

3.3.2 Dialogue Management

Dialogue State Machine The primary component responsible for dialogue management within E-STDM is a state machine. In the framework, state transitions alternate between the user and the system to track turn taking, and are defined by NATEX (Figure 3.1). Any transition performed with a slot-filling NATEX will store a slot-value pair in a dedicated table in memory, which persists globally for future reference.

User turns are modeled by transitions according to which NATEX matches the user input. To resolve cases where multiple NATEX yield matches, transitions can be defined with priority values. Similarly, developers can specify a catch-all “error transition” (ERROR in Figure 3.1) to handle cases where no transition’s NATEX returns a match. The user input resulting in an error transition is automatically logged to improve the ultimate design of the state machine.

System turns are modeled by randomly selecting an outgoing system transition. Random selection promotes uniqueness among dialogue pathways, but can be restricted by specifying explicit priority values. To avoid redundancy when returning to a previously visited state, E-STDM prefers system transitions that have not been taken recently.

The simplicity of this dialogue management formulation allows for rapid development of contextually aware interactions. The following demonstrates the streamlined JSON syntax for specifying transitions S_1 , S_3 , U_1 , U_2 , and U_3 in Figure 3.1.

²<https://github.com/lark-parser/lark>

```

{
  "Have you seen any movies lately?": {
    "state": "c",
    "[I, $ENT=#ONT(entertainment)]": {
      "What's your favorite $ENT?": {...}
    },
    "$MOVIE=#MDB()": {
      "$MOVIE is one of my ...": {...}
    }
  },
  "error": {
    "Sorry, I didn't catch ...": "c"
  }
}

```

Information State Update Rules Despite its simplicity, state machine-based dialogue management often produces sparsely connected state graphs that are overly rigid for complex interactions [34]. E-STDM thus allows developers to specify information state update rules to take advantage of the power of information state-based dialogue management.

Information state update rules contain two parts, a precondition and a postcondition. Each user turn before E-STDM's state machine takes a transition, the entire set of update rules is iteratively evaluated with the user input until either a candidate system response is generated or no rule's precondition is satisfied. In the following example, satisfying precondition `[I have $USER_PET=#PET()]` triggers postcondition `#ASSIGN($USER_LIKE=$USER_PET)` to assign `$USER_PET` to `$USER_LIKE`, allowing rule `#IF(..) I like $USER_LIKE ..` to trigger in turn:

```

{
  "[I have $USER_PET=#PET()]"
  : "#ASSIGN($USER_LIKE=$USER_PET)",
  "[$USER_FAVOR=#PET() is my favorite]"
  : "#ASSIGN($USER_LIKE=$USER_FAVOR)",
  "#IF($USER_LIKE != None)"
  : "I like $USER_LIKE too! (0.5)"
}

```

When a precondition is satisfied, the postcondition is applied through the language generation (Sec. 3.3.1). If a real-number priority is provided in parentheses at the end of any NATEX, the generated string becomes a candidate system response. A priority value higher than any outgoing system transition in the dialogue state machine results in the candidate becoming the chosen one; thus, no dialogue state machine transition is taken. Often however, a developer can choose to omit the priority value to use NATEX purely as a state updating mechanism.

This formalism allows flexible interoperability between state machine-based and information state-based dialogue management. Given E-STDM, developers have the latitude to develop a system entirely within one of the two approaches, although a mixed approach lends the best balance of development speed and dialogue sophistication.

3.4 Design and Deployment of Emora

This section presents Emora, a socialbot developed in collaboration with the Emory University Alexa Prize 3 team. The goal of Emora's development and deployment was to provide Amazon Alexa users with engaging human-computer dialogue in the open-domain chat-oriented setting of the Alexa Prize. To accomplish this, a major design goal of Emora was to create interesting and fluent dialogue, but to reach beyond information-based chat so that Emora feels more like a caring friend to users than an information desk. In addition to providing a valuable application to users, Emora's design tests the following hypothesis: conversational AI is more engaging to users when

focusing on personal chat that prioritizes opinions and experiences instead of information-oriented chat concerned with providing detailed and accurate information.

Section 3.4.1 describes the user experience design and philosophy of Emora’s development in additional detail. Section 3.4.2 provides an analysis of Emora’s performance in the Alexa Prize application setting.

3.4.1 Conversation Design

Following the goal to design Emora as a chatbot capable of engaging in human-like social aspects of conversation, Emora was designed to reach beyond opinion-oriented conversation and support personal conversation. Personal conversation is distinguished from opinion-oriented conversation based on whether speakers are sharing thoughts and attitudes towards entities that are regularly present and impactful to their daily lives. For example, stating an opinion about a movie would be regarded as opinionated but not personal, whereas a speaker expressing an opinion towards a class they are enrolled in would be considered personal, since the class has a direct and substantial involvement in the speaker’s own life. Although this distinction exists along a continuum, it is helpful to explicitly focus on driving a more personal conversation to improve the human-like qualities of Emora.

Table 3.1 presents a simulated conversation with the socialbot. The given conversation is an exemplification of the personal-experience-focused dialogue strategy that is the main novelty of the approach. It provides the general interaction style of sharing and receiving personal experiences with the user in a manner that balances displaying an interest in the user’s experiences through question-asking and giving open-ended statements that allow the user the freedom to respond as they wish.

Supporting personal, experience-oriented conversation in open-ended chat is especially challenging for two reasons. First, experience-oriented conversation evokes a wider range of user satisfaction than information-oriented conversation. Engaging with users on subjects such as their daily experience, recent challenges, and interpersonal relationships can lead to deeper, more meaningful

ID	Utterance	Slot Table	Stack
E1	I've been hearing a lot from people about how strange it is to live with the corona virus. Has your life changed much?		[]
U1	Yeah my school has online courses now	student=True remote=True	[CoS]
E2	Oh, are you liking your online classes?		[CoS]
U2	Not really		[CoS]
E3 (a)	Oh yeah, it is hit and miss for a lot of people. Definitely just try to do the best you can.		[CoS]
E3 (b)	Actually, I have seen something pretty uplifting recently. Have you seen those videos where zoos are letting some of their animals out of their cages to roam around?		[]
U3	No		[]
E4 (a)	My favorite one was the penguins at the Chicago zoo, you should check it out!		[]
E4 (b)	Anyway, the good news is that this virus won't last forever and people are taking steps in the right direction to lower its impact. I know things might seem weird right now, but just do the best that you can and stay positive.		[]
U4	I will try to but its pretty stressful, i am having a hard time buying groceries	need_grocery=True	[CoE]
E5	Yeah, that's tough. None of my friends can find necessities like toilet paper in stores anywhere. Honestly, who would have ever imagined there would come a time when toilet paper was such a sought after item?		[CoE]
U6	It is crazy		[CoE]
E6 (a)	For sure.		[CoE]
E6 (b)	So, what did you get up to today?		[]
U6	I called my mom	activity=talk_to_mom	[]
E7	It is good to hear that you are keeping in touch with the people in your life. Are you close to your mom?		[]
U7	We are pretty close		[]
E8 (a)	Wow, you sound like you are really close to your family. I am glad to hear that.	close_to_mom=True	[]
E8 (b)	So, I remember you saying something about being a student. I am actually a student too. Do you like school?		[]
U8	Yes I do	likes_school=True	[]

Table 3.1: Simulated interaction with Emora, modeled on interactions observed from real users. User inputs are denoted as U#. System responses involving multiple state transitions are denoted on separated contiguous lines as E# [a-b]. The third column indicates the slot values tracked from a given user utterance. The fourth column displays the contents of the stack at each point in the conversation (CoS: covid_sympathy, CoE: covid_end).

interactions, but it can also quickly introduce negative emotions and frustration if these topics are not handled carefully. A second challenge is that experience-oriented conversation often requires reasoning abilities. For example, suppose the user mentions a spouse to the socialbot. In this case, if the socialbot then asks the user "Are you a student?", the socialbot will likely be perceived as inattentive or unintelligent by the user, often provoking a reaction of frustration. On the other hand, the socialbot's ability to successfully reason and provide a response that incorporates information shared by the user, such as responding "Do you have any kids?", will often result in an engaging experience where the user feels understood.

The E-STDM approach used to design Emora makes it easy to encode custom topic introductions and dynamic commonsense reasoning, since slots encoding what Emora has learned about the user are stored in the structured conversation state as the conversation progresses. By handcrafting responses and custom policy rules that are dynamically triggered based on Emora's dialogue state representation, Emora leverages the commonsense reasoning ability of human developers to make engaging interactions more likely and avoid frustrating ones.

3.4.2 Results and Analysis

Impact of Opinion-Oriented Conversation Design Since one of the major focuses for the socialbot was on incorporating opinion-driven conversations, empirical results are presented quantifying the importance of opinion sharing in chat. Over a period of three days in early April, an A/B test was conducted between a fact-oriented coronavirus opening and an opinion-oriented coronavirus opening, keeping the remainder of the system stable. Half of the users received the fact-oriented opening whereas the other received the opinion-oriented opening. Table 3.2 shows the results of the A/B test, suggesting that opinion-oriented conversation is better received among users, even when the broader conversation domain is the same.

Impact of Personal Conversation Design In addition to the analysis of the opinion-oriented conversation, an analysis is performed to investigate the impact of conversation that focuses

Type	Average Rating
Fact	3.59
Opinion	3.73*

Table 3.2: Results of A/B test for fact-based and opinion-based coronavirus openings. (*) denotes a statistically significant result, $p < 0.10$.

specifically on personal experiences and information. The average rating was calculated for conversations that contained the *Life* topic³ and the average rating of conversations that contained a conversational style that did not focus on personal experience sharing: *Movies*, *Music*, *Virtual Reality*, and *Sports*. The results of this analysis appear in Table 3.3. Note that, as described in Section 3.4.1, personal conversation is distinguished from opinion-oriented conversation that does not contain personalized elements. The four above components whose ratings comprise the "Info and Opinions" category in Table 3.3 have an opinion-oriented focus, but do not focus on personal information. Other topics are excluded from the analysis, such as the pets component, since it has some elements of personal information sharing as well as many fact-oriented elements. Additionally, conversations with 3 or less turns were removed to reduce noise from uncooperative users. Results indicate that experience-oriented conversations are preferred to opinion-oriented conversations.

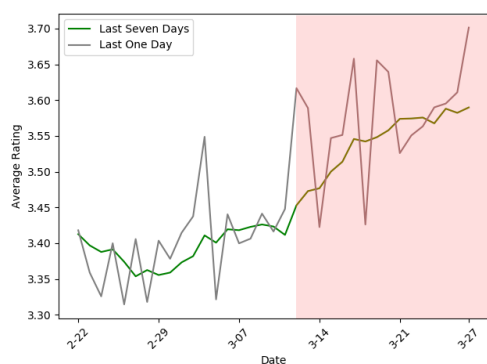
Type	Average Rating
Info and Opinions	3.79
Personal Experience	3.85*

Table 3.3: The average rating of conversations stratified by the types of components they contain. (*) denotes a statistically significant result, $p < 0.10$.

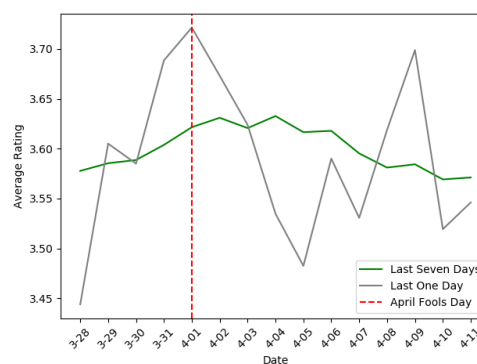
Impact of Current Events on Conversation Satisfaction In order to begin conversations with users in a reliably natural way, several conversation openings were crafted that specifically targeted holidays and current events. For Valentine’s Day, April Fool’s Day, and Easter, these opening topics were deployed instead of a generic opening. Figure 3.2 shows the average user score over the time periods where these current-event-based openings were deployed. the results present an indication

³Mutually-exclusive topic labels were assigned to each of Emora’s state nodes in the dialogue transition graph, making topic mentions measurable.

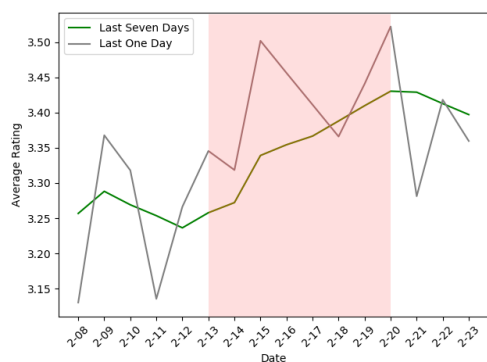
that these special openings have a positive impact on the user’s satisfaction and impression of the socialbot. All time periods in which the Holiday openings were deployed display a positive trend in average user ratings. Similar to the Holiday openings, the COVID-19 opening shown in Figure 3.2(a) also displays a positive effect on the user ratings. Unlike the Holiday components, whose underlying current event is more transient, COVID-19 was a dominating event during Emora’s deployment and, in line with this, the positive impact of the COVID-19 Opening seems to have been sustained over time.



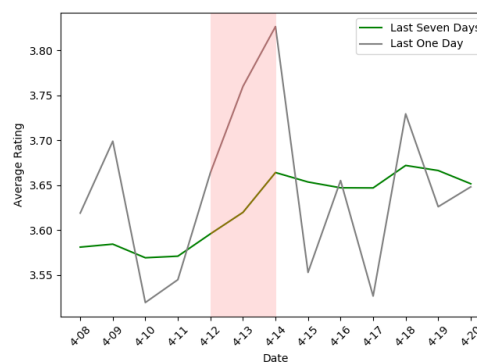
(a) The period when COVID-19 was the opening.



(b) The day that April Fools Day was the opening.



(c) The period when Valentine’s Day was the opening.



(d) The period when Easter was the opening.

Figure 3.2: The average daily and last seven day user ratings for the time periods surrounding the current-event-focused openings. Single day openings are indicated by a red dashed line and multi-day openings are denoted by the red highlighted section in each graph.

Overall Performance Figure 3.3 shows the daily and last seven day average user ratings that were received during the course of the Quarterfinals and Semifinals periods. There is a definite positive trend, providing empirical support that the various changes and system updates made to

Emora’s dialogue management improved user experience. Additionally, Emora took first place in the final competition results out of the 10 socialbots submitted by university teams. Final placement was decided by third-party judges invited by Amazon who scored Emora an average quality of 3.81 out of 5 on a Likert scale. The second and third place runners-up scored 3.17 and 3.14, respectively.

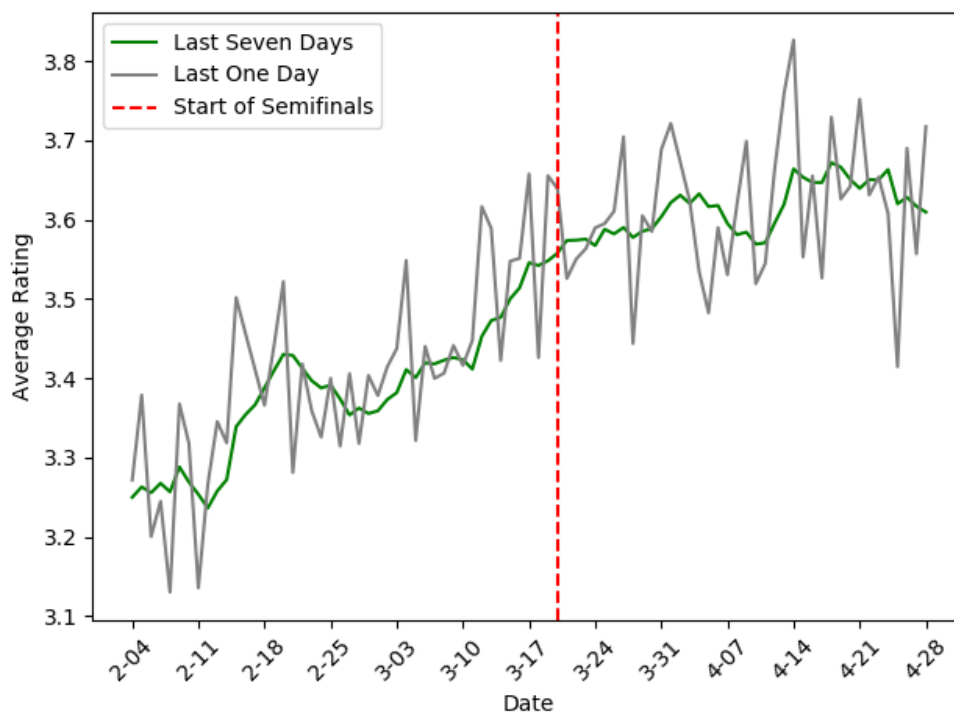


Figure 3.3: The daily average user rating and the average seven day user rating received during the Quarterfinals and Semifinals.

3.5 Experimental Evaluation of Emora

The Emora-STDM approach to human-computer dialogue is radically different than the modern approach seen in state-of-the-art research. This is primarily due to the design philosophy of Emora-STDM for supporting flexible and controllable real-world dialogue applications, at the expense of a large amount of developer effort required to handcraft NLU NATEX patterns and response logic. By contrast, research focuses on data-driven approaches for human-computer dialogue. These approaches are in theory more scalable since they are capable of creating dialogue systems at

virtually zero human labor cost for any scenario in which data is available. However, training a neural human-computer dialogue model in a way that only optimizes the objective of imitating human behaviors seen in the data fails to account for the unavoidable need to control dialogue system behavior if the goal is to deploy the dialogue system in a live application setting. Thus, a question arises: what are the characteristic differences in the behavior and overall performance of handcrafted vs state-of-the-art data-driven approaches to dialogue system development? This section addresses this question by presenting an experimental evaluation of Emora, a case study of a dialogue system relying entirely on handcrafted elements operating atop structured dialogue state, as well as three of Emora’s contemporaries that claimed state-of-the-art for various aspects of chat-oriented dialogue.

3.5.1 Chatbot Selection

To evaluate the strengths and weaknesses of chat-oriented dialogue (COD) models, chatbots are selected using a two-stage process: (1) a literature review to identify chatbot candidates, and (2) a pilot evaluation to select the final set of bots.

Literature Review To promote diversity among the selected chatbots, the review focused on four popular themes of the human-computer chat: (1) Knowledge-grounded chat, (2) Empathetic chat, (3) Self-consistent chat, and (4) General open-domain chat with large pre-training resources like Reddit. Candidate chatbots are selected from each theme using the following criteria:

1. The bot must demonstrate state-of-the-art performance in a task related to the theme.⁴
2. The implementation must be provided.
3. The response latency of the bot must be <10 seconds using modern GPU hardware.

This review yields the 6 chatbot candidates in Table 3.4: Blender-Decode [49], Blender2 [82], BART-FiD-RAG [70], Emora [14], DukeNet [45], and CEM [66].

⁴Note that selection occurred in October 2021.

Model	Theme	N	Q	Pass
Blender-Decode	Consistency	10	4.1	✓
Blender2	General	10	3.8	✓
BART-FiD-RAG	Knowledge	10	3.5	✓
Emora	General	10	3.3	✓
DukeNet	Knowledge	9	1.9	✗
CEM	Empathy	12	1.1	✗

Table 3.4: The pilot results for 6 bots, showing the theme of the approach (**Theme**), the number of collected conversations (**N**), and the avg. dialogue-level Likert quality rating (**Q**). **Pass** denotes which models passed the verification criteria and were included in the full study.

Chatbot Selection A pilot evaluation using the 6 chatbot candidates is conducted in order to verify the multi-turn dialogue capability of the chatbot candidates. 10 students majoring in Computer Science or Linguistics are invited to interact with randomly assigned chatbots in 3-5 text-based conversations, each of which consisted of 30 turns.⁵ At the end of each conversation, students are asked to rate the quality from 1 (least) to 5 (most). Based on the pilot results (Table 3.4), DukeNet and CEM are excluded from the full evaluation because they are unable to hold satisfying multi-turn conversations, despite their reasonable single-turn response generation capabilities.

3.5.2 Evaluation Methods

Two human evaluation methods are chosen for the study, both of which ask a 3rd party human judge to read the text conversation between a chatbot and human interactor and follow instructions to provide judgement labels.

Turn Likert Annotators provide turn-level ratings from 1 (least) to 5 (most) for the 8 labels shown in Table 3.5. The label set proposed in Finch and Choi [13] is used, which results from a detailed survey of characteristics used in chat evaluation and has better coverage than alternatives like the set used in ACUTE-Eval [36]. The dialogue-level metric is measured as the mean rating of a single dialogue’s turns. The bot-level metric is calculated as the mean rating of all turns.

⁵A “turn” is defined as ONE message from a single interactor.

Label	Abbr.	Definition
Consistency	Con_t	Responses do not produce information that contradicts other information known about the system
Emotion Understanding	Emo_t	Responses indicate an understanding of the user’s current emotional state and provide an appropriate emotional reaction based on the current dialogue context
Engagingness	Eng_t	Responses are engaging to the user and fulfill the particular conversational goals implied by the user
Grammaticality	Gra_t	Responses are free of grammatical and semantic errors
Informativeness	Inf_t	Responses produce unique and non-generic information that is specific to the dialogue context
Quality	Qua_t	The overall quality of and satisfaction with the dialogue
Proactivity	Pro_t	Responses actively and appropriately move the conversation along different topics
Relevance	Rel_t	Responses are on-topic with the immediate dialogue history

Table 3.5: The 8 labels for Likert evaluations, henceforth referred to using their abbreviations and colors.

Behavior Classification: ABC-Eval is a novel evaluation methodology for chat-oriented dialogue in which annotators provide binary labels on the turn-level indicating the presence or absence of a particular chat characteristic. The included chat characteristics are defined in Table 3.6. Dialogue-level metrics are calculated as the proportion of turns that display the characteristic of the dialogue. Bot-level metrics are calculated as the proportions of turns that display the characteristic over all bot dialogues. ABC-Eval labels were validated as being more reliable and informative regarding dimensional aspects of chat quality as opposed to existing evaluation methods, such as average Likert ratings [15].

3.5.3 Evaluation Study

The full evaluation study consists of the collection of 24 labels per conversation. This collection was split into 16 independent evaluation tasks as follows:

- 8 ABC-Eval tasks, each composed of 1 to 4 labels as denoted by groupings in Table 3.6
- 8 Turn Likert tasks, each composed of 1 label from Table 3.5

Label	Abbr.	Description	Inspired by
Uninterpretable	!Int_b	It is difficult to understand the intended meaning of part or all of the response.	1, 2, 3, 4, 5, 6
Antisocial	!Soc_b	The response is insulting, hateful, or excessively vulgar.	2, 7, 8, 9
Preference Info Life Info	Pre_b Lif_b	The response expresses the bot’s preferences, wishes, or values. The response shares information about the bot’s life or experiences.	10, 11
Empathetic Lack of Empathy	Emp_b !Emp_b	The response shows an understanding and reacts appropriately to someone’s emotions. The bot misunderstands or reacts inappropriately to someone’s emotions.	11, 12, 13
Commonsense Contradiction	!Com_b	The response misunderstands or contradicts common knowledge.	2, 14, 15, 16
Fact Usage Fact Contradiction	Fac_b !Fac_b	The response accurately incorporates encyclopedic or expert knowledge. The response hallucinates or inaccurately presents encyclopedic or expert knowledge.	1, 2, 11, 17, 18, 19, 20
Self Contradiction Partner Contradiction Redundant	!Sel_b !Par_b Red_b	The bot contradicts something it said earlier in the dialogue. The bot contradicts or misremembers something the user said earlier in the dialogue. The response inappropriately repeats information presented earlier in the dialogue.	2, 3, 6, 20, 21, 22, 23
Ignore Irrelevant Follow-up Topic Switch	Ign_b !Rel_b Fol_b Top_b	The response ignores what the user just said. The response interrupts the current topic of discussion by presenting unrelated information. The response explores, elaborates on, or asks about the ideas shared in the previous turn. The response introduces a new topic of conversation.	1, 2, 3, 6, 24

Table 3.6: The 16 behavior labels within ABC-Eval. Row separators denote evaluation task groupings. **Bold** indicates behavior labels kept in final set. [1] Gopalakrishnan et al. [17], [2] Higashinaka et al. [23], [3] Mehri and Eskenazi [42], [4] Mehri and Eskenazi [43], [5] Phy et al. [53], [6] Sanguinetti et al. [67], [7] Beattie et al. [3], [8] Sun et al. [73], [9] Xu et al. [87], [10] Rashkin et al. [61], [11] Smith et al. [71], [12] Majumder et al. [40], [13] Rashkin et al. [60], [14] Zhong et al. [95], [15] Zhou et al. [96], [16] Zhou et al. [97], [17] Gupta et al. [18], [18] Honovich et al. [24], [19] Santhanam et al. [68], [20] Shuster et al. [70], [21] Li et al. [37], [22] Nie et al. [49], [23] Welleck et al. [80], [24] Xu et al. [88] .

The 16 evaluation tasks are posted on SurgeHQ’s annotation platform⁶ to be completed by dedicated remote workers (Surgers) with experience in NLP annotation. Each time an evaluator connects to one of the evaluation tasks, they are assigned a randomly selected conversation to annotate. Evaluators are compensated per annotated conversation per task, at an estimated rate of \$20/hr. Evaluators are allowed to annotate up to 60 conversations per task. The final evaluation dataset consists of 400 conversations, each with results for all 24 labels.

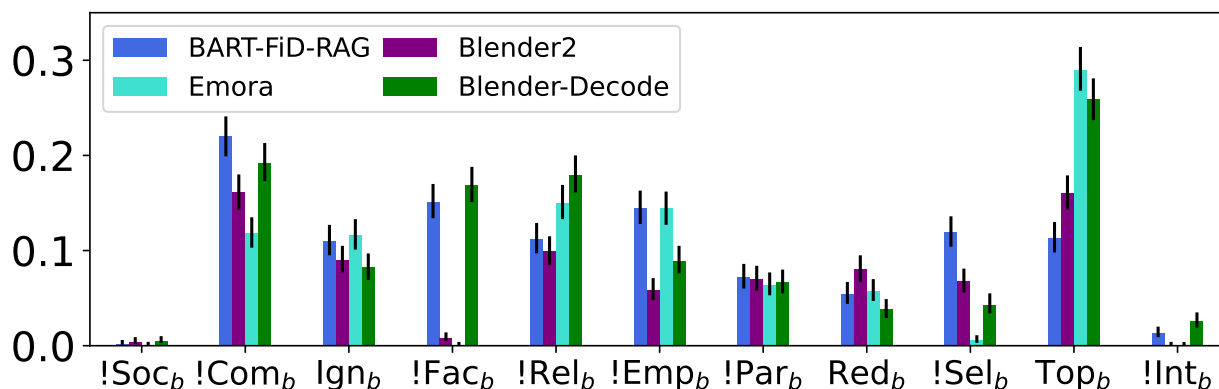


Figure 3.4: Proportions of turns expressing undesirable behaviors, with 95% Wilson score confidence intervals.

3.5.4 Chatbot Evaluation

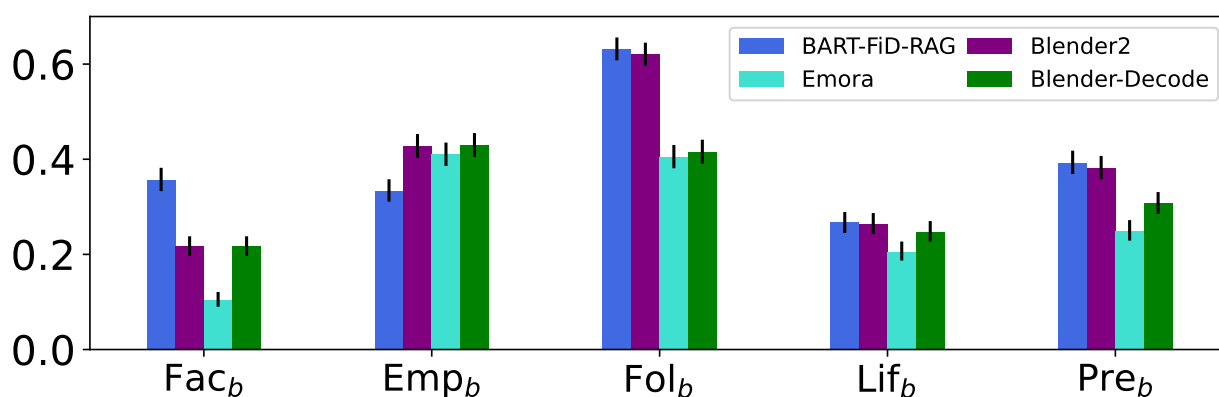


Figure 3.5: Proportions of turns expressing desirable behaviors, with 95% Wilson score confidence intervals.

To evaluate the strengths and weaknesses of our 4 selected chatbots, results are presented for the 400 collected conversations across all ABC-Eval metrics (Figure 3.4 and Figure 3.5), and Likert Turn metrics (Figure 3.6).

Commonsense Violations of commonsense are one of the most frequent problems reported across the 4 bots, with about 10-20% of turns containing a commonsense contradiction, depending on the chatbot. Emora demonstrated the best commonsense ability of all chatbots.

⁶<https://www.surgehq.ai>

Coherence All chatbots had occasional issues maintaining dialogue coherence. Across bots, about 10-15% of responses contained irrelevant information, and around 10% of responses ignored something the human interactor said completely. Based on our results, Emora had the overall worst coherence capability, performing poorly on Rel_d and Rel_t , worst on the ignore error Ign_b , and 2nd worst on the irrelevant error $!Rel_b$.

Consistency Although the rate of partner contradictions, self contradictions, and redundancies was relatively lower than some other issues like ignoring the user or commonsense violations, these consistency issues each occurred about 5-10% of the time in the three data-driven models. True to its approach, Blender-Decode was able to mitigate all three inconsistent behaviors compared to the other two data-driven bots; however, its poor performance on other ABC-Eval metrics suggests that incorporating the Decode model into response generation created trade-offs between consistency and other aspects of dialogue quality. Unsurprisingly, Emora had virtually no self-contradictions $!Sel_b$, although redundancies Red_b and partner contradictions $!Par_b$ occurred about as often as with the neural chatbots.

Knowledge Rates of correct fact usage show that BART-FiD-RAG is the most factually-oriented chatbot, with nearly 40% of its responses containing correct factual world knowledge, compared to about 20% in the other data-driven models. However, BART-FiD-RAG also presents incorrect factual information in about 15% of its responses, along with Blender+Decode. These rates of presenting incorrect information pose a serious challenge to settings where misleading information is unacceptable. Notably, Blender2 had a very low ($< 1\%$) rate of presenting incorrect information, even though it incorporated world knowledge in 20% of its responses. Emora never hallucinated factual knowledge due to her handcrafted response generation approach.

Empathy All four bots seem to appropriately respond to a their human partner’s emotions at high rates (about 30-40% of turns). However, there is still room for improvement, since our evaluation shows that chatbots failed to give an emotionally appropriate response 5-15% of the time. Blender2

was the best overall at understanding and reacting to the interactor’s emotions, with only 5% of its turns labeled with $!Emp$. Emora’s empathetic ability was poor, with about 15% $!Emp_b$ errors.

Overall Quality Shown in Figure 3.6, our overall quality results suggest the following ranking: Blender2 > Emora > Bart-FiD-RAG > Blender-Decode . These quality results are consistent with the rates of erroneous behaviors we see among these chatbots.

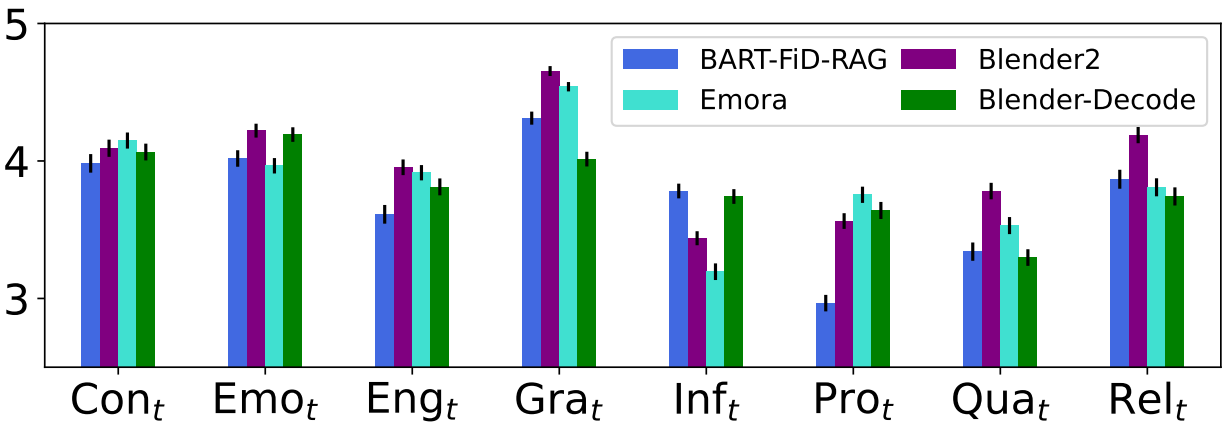


Figure 3.6: Average Turn Likert ratings of the conversations, with 95% Student’s t confidence intervals.

3.6 Discussion

Overall, the results demonstrate that the characteristics of Emora and neural chatbots are quite distinct, despite the overall quality of Emora being competitive with even the best neural chatbot Blender2. In particular, neural chatbots were able to respond more *appropriately* in terms of the relevance and empathy of their responses, with fewer instances of ignoring the user compared to Emora. On the other hand, Emora is more *consistent*, with virtually zero self-consistency issues or factual knowledge hallucinations, and superior commonsense ability compared to neural bots.

These results give some empirical backing to the theory that rule-based and neural systems are complimentary in their strengths and weaknesses. While Emora is incapable of responding appropriately to any dialogue context due to a reliance on handcrafted DST patterns and response templates,

these constraints also enforce more reliable dialogue interactions in many cases. Furthermore, the results are a demonstration of the effectiveness of handcrafted DST and dialogue policies, as Emora was able to achieve competitive performance with the SoTA neural models. On the other hand, the fact that Emora’s response generation relies entirely on handcrafted elements is a clear weakness, as demonstrated by a high rate of relevance and appropriateness issues with her responses.

One of the most straightforward ways to combine the strengths of neural and rule-based approaches demonstrated in this study is to simply augment neural systems with a structured dialogue state representation using a dialogue state tracking model. Even if response generation is not conditioned on the structured dialogue state directly, the presence of an accurate dialogue state at each turn in the conversation would allow integrations with external components and an easy mechanism to trigger custom behaviors when certain conditions are met. For example, large language model (LLMs) prompting [50] provides a simple mechanism with which responses can be reliably conditioned by custom policy controls triggered by conditions in a structured dialogue state representation.

Despite the apparent utility, a major challenge for incorporating SoTA DST models into open-domain dialogue systems is the limited nature of current DST data. Virtually all DST data targets closed-domain, task-oriented systems, and suffer from a lack of variety in slot types. Future work should look to improve the domain flexibility of DST, particularly in settings where only small adaptation resources are required like zero- and few-shot DST, for usage in open-domain and chat-oriented systems. The next chapter presents a strong start towards this direction by vastly increasing the domain coverage and generalizability of dialogue state inference.

Chapter 4

Dialogue State Generation

Conventionally, Dialogue State Tracking (DST) relies on a predefined slot schema specifies the types of information (slots) to be tracked. This slot schema is provided to DST models in the form of labeled examples or natural language descriptions for each slot to perform domain adaptation. However, defining the slot schema and collecting these domain adaptation resources is costly [62, 5], resulting in a barrier to apply DST in practice. Furthermore, the domain adaptation abilities of DST models may be limited due to poor diversity in training data caused by the high cost of data collection. To alleviate these resource barriers, the Slot Schema Induction task has been proposed to automatically induce a slot schema and value examples from unlabeled dialogue data [27, 56, 85]. However, Slot Schema Induction approaches rely on clustering value spans extracted from a large amount of unlabeled dialogue data, which may not be available for many application settings. Furthermore, outputted value clusters contain no slot names and require substantial human interpretation and revision to finalize the slot schema.

This chapter presents a new task, Dialogue State Generation (DSG), which aims to infer dialogue states for new domains while eliminating the resource constraints imposed by both DST and Slot Schema Induction. Given a dialogue context, DSG models are required to infer both the slots and values of the dialogue’s state without access to *any* domain-specific resources such as slot names, descriptions, examples, or other unlabeled dialogues. Thus, success in this task would provide

reliable dialogue state tracking in virtually any domain without the need to define a slot schema or collect other resources, since slot schema induction occurs automatically as a part of the model’s inferences.

An approach and evaluation are presented that tackle the main initial challenge for DSG: domain generalizability. Because DSG models are provided zero in-domain resources at inference time, they must be able to generalize to any domain to discover appropriate slots and fill them with correct values. To achieve this, supervised training of a DSG model would thus require a high degree of training data diversity; however, existing DST datasets cover a limited number of domains. This challenge is overcome by leveraging large language models (LLMs) to construct a domain-general DSG pipeline using prompting instead of fine-tuning. Due to the steep inference cost of LLMs, a model is also distilled¹ from the LLM pipeline by generating a synthetic dialogue dataset spanning 1,000+ domains, silver-annotating it with the pipeline, then training an end-to-end DSG model. Experiments demonstrate that our DSG models successfully generalize to new domains and out-of-distribution data with negligible performance drop-off, whereas a baseline model trained on existing data is incapable of such generalization. The presented DSG task and models thus make substantial progress towards zero-resource dialogue state inference.

4.1 Dialogue State Generation (DSG)

Dialogue State Generation (DSG) task is defined as follows: given a dialogue history $D_{1..t}$ from turn 1 to t , it aims to produce a set of slot-value pairs $S_t = \{(s_1, v_1), (s_2, v_2), \dots, (s_k, v_k)\}$, which accurately represents all information in $D_{1..t}$ relevant to either speaker’s goals. Unlike DST where a specification of goal-related slots is predefined by the slot schema, DSG requires discovering slots that are appropriate for the dialogue, in addition to correctly identifying their values. Other than the dialogue history itself, no guidance or resources are provided to the DSG model during inference. Moreover, in this work, DSG models are exclusively tested on unseen domains, which prevents

¹Using symbolic distillation as in West et al. [81]

signaling in-domain information via training examples.

Accurate DSG inferences are characterized by the *Correctness* of each slot-value pair (s_i, v_i) in representing the type and value of some information appearing in $D_{1..t}$, and the *Completeness* of each inferred state S_t in covering all goal-related information appearing in $D_{1..t}$. Additional definition of *Correctness* and *Completeness* are given in Section 4.6.3. Note that since the primary goal of this work is to tackle the problem of domain generalizability for DSG, this work does not evaluate whether slot names maintain consistency across multiple inferences. Although the consistency of slot naming is required to link DSG slot outputs to application infrastructure such as API call parameters, this problem is partially solved via simple pattern matching or clustering to resolve slot names, and it is left to future work to explore a full solution.

Finally, note that the models presented in this work infer each state *update* U_t from $D_{1..t}$, which represents the set of slots whose values have changed since the previous turn such that:

$$S_t = \text{update}(S_{t-1}, U_t)$$

This results in $O(|U_t|)$ for obtaining S_t from S_{t-1} , which would be $O(|S_t|)$ if all dialogue states in S_t were generated individually ($|U_t| \leq |S_t|$).

4.2 Related Work

Dialogue State Tracking is the task of inferring the correct value for each of several predefined slots throughout the course of a dialogue. DST has been tackled using fully supervised [20, 86, 83] and few-shot [39, 69, 7] approaches, where the model learns to track each slot using labeled example data. The utility of these approaches is limited in practice because of their reliance on labeled examples, which are costly to collect. Zero-shot DST addresses the cost of data collection by providing models with only a natural language description for each slot [30, 22, 78, 75]. However, zero-shot DST still incurs a nontrivial labor cost to work, especially since the development of slot descriptions often requires several revisions before the developer’s intent is aligned with the

model interpretation of each description. DSG eliminates this labor cost by requiring the model to infer all relevant dialogue state information without any predefined resources. In this way, model interpretations are all provided “up-front,” allowing developers to immediately select useful slots from model inferences to be integrated into a broader dialogue application.

Slot Schema Induction is the task of extracting important information from unlabeled dialogue data and separating these extracted information values into appropriate slot categories. Like DSG, Slot Schema Induction must infer what slots and values are critical for the dialogue’s goals with no guidance other than the dialogue text itself, and was proposed to alleviate the manual effort of slot schema creation. Recent approaches to this task leverage tagging models trained on related tasks such as semantic role labeling to identify candidate values, then cluster vector encodings of these values into slots [27, 56, 85, 92]. There are three major limitations of Slot Schema Induction that DSG addresses. First, Slot Schema Induction assumes access to a large number of unlabeled dialogues within a particular task domain in order to induce appropriate slots, whereas DSG models are capable of discovering important slots from even a single dialogue. Second, Slot Schema Induction is limited to discovering slots and values as extracted spans from the original dialogue text; DSG’s values can instead be abstracted or inferred. Finally, Slot Schema Induction treats each value cluster output as an induced slot, which requires manual effort to interpret and name each cluster to finalize the slot schema. On the other hand, DSG models infer slot names as a part of the DSG task formulation, making discovered slots immediately interpretable.

DST Data Diversity One of the challenges facing recent work in DST is the limited amount of available training resources, particularly with respect to the diversity of task domains and slot types. The most diverse dataset, Schema Guided Dialogues (SGD), covers 16 domains and 214 slot types [62]. MultiWOZ covers 5 domains and 31 slot types [5, 89]. The low diversity in these training resources is likely to limit generalizability to unseen target domains. Some existing works in few-shot DST mitigate this issue by generating synthetic dialogues for the target domain using template-based methods [2, 1] or PLMs [29, 48, 77]. However no previous work has attempted fully

automatic generation of new task-oriented dialogue domains. DSG addresses the limited diversity in existing resources by automatically annotating a synthetic dataset of over 1,000 task domains with silver dialogue state labels.

4.3 GPTPipe: DSG with Zeroshot Pipeline

The primary challenge of DSG is the generalization of dialogue state inference to diverse information types and domains. Given the exceptional zero-shot performance of instruction-tuned large language models (LLMs) on a wide variety of tasks [4, 31, 22], they are leveraged as an initial approach to DSG. Specifically, this section presents a pipeline that uses LLMs to automatically annotate every dialogue turn with a state update in two stages: *Question-Answer (QA) Pair Generation* to deduce the key information in each turn (Section 4.3.1) and *Slot-Value Translation* to transform those QA pairs into the corresponding slot names and values (Section 4.3.2). Figure 4.1 illustrates the full pipeline approach. Since GPT² models are selected as LLMs for our experiments, the approach is denoted as GPTPipe.

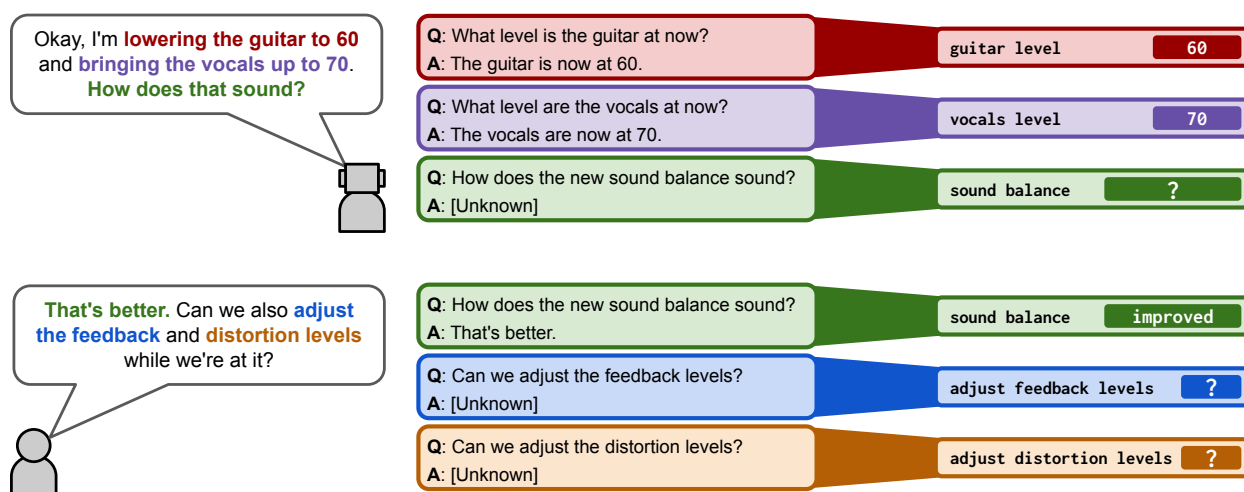


Figure 4.1: Two example inferences of GPTPipe, a GPT-based DSG pipeline that infers dialogue state information by first generating question-answer pairs for important information shared in a given dialogue turn (Section 4.3.1), then translates questions to slot names and answers to values to obtain the final dialogue state update (Section 4.3.2).

²gpt-4-0314 and gpt-3.5-turbo-0301 were used for QA Pair Generation and Slot-Value Translation respectively.

4.3.1 Question-Answer Pair (QA) Generation

To generate a state update U_t given a dialogue history $D_{1..t}$, a prompt P_t containing the last two turns $D_{t-1,t}$, instructs GPT to break down all the information in turn t as a set of QA pairs. Only the last two turns are included to reduce irrelevant information from previous turns that could misguide the state update for the current turn t . To further mitigate this issue, every turn is prepended with a speaker tag, allowing GPT to solely focus on turn t by referring to the corresponding speaker. A set of QA pairs $QA_t = \{(q_1^t, a_1^t), \dots, (q_k^t, a_k^t)\}$ is generated by this method, where each question q_i^t represents an information type either shared or requested during the turn and its answer a_i^t summarizes the information value.

State updates are produced to monitor the change in values of slots throughout the dialogue, enabling tracking whether information requests from one speaker are satisfied through information shared by the other speaker. To implement this, P_t explicitly designates the answer *Unknown* for use in any QA pair, where the question represents an information request made by the current speaker. Therefore, for each turn, a set of unanswered questions for the prompt P_t can be identified as follows:

$$R_t = \{\forall_i. q_i^t : 0 < i \leq t \wedge a_i^t = \text{Unknown}\}$$

A second prompt P'_t is used to answer each question in R_t using two turns $D_{t,t+1}$, which produces a set of QA pairs QA'_{t+1} comprising slots from turn t filled with values in turn $t + 1$. Included in P'_t is an instruction to use *Unknown* for questions whose answers are not present in turn $t + 1$. Such unanswered questions are removed from QA'_{t+1} , leaving only QA pairs with information requested in turn t and shared in turn $t + 1$. QA'_{t+1} are then appended to the next prompt P_{t+1} to generate a new set QA_{t+1} for turn $t + 1$. Including QA'_{t+1} in P_{t+1} guides GPT to generate only new QA pairs that have not already been covered by QA'_{t+1} .

4.3.2 Slot-Value Translation

The second stage of `GPTPipe` translates every QA pair in QA_t (Section 4.3.1) to a slot-value pair. GPT tends to generate overly detailed slot names when answers are provided along with questions. Hence, slot names and values are derived using separate prompts. First, a prompt P_s is used to translate all questions in QA_t into corresponding slot names. No context from the dialogue is provided, nor do we include any answers from QA_t in P_s . The result is a set of slot names $N_t = \{s_1^t, \dots, s_{|QA_t|}^t\}$ representing information types mentioned in turn t .

Finally, a prompt P_v , comprising questions and answers in QA_t as well as the slot names in N_t , is used to translate each answer into a value for the corresponding slot name. In addition, P_v highlights that a value can be a concise phrase, number, span, category, score, boolean, list, or other form, aiding the model in generating values suitable for the respective slot names, rather than always using natural language phrases as values. QA pairs with the *Unknown* answer are excluded from P_v , as they are translated into a special token `?` to represent a requested slot. Pairing each generated value with its corresponding slot name results in the dialogue state update $U_t = \{(s_1^t, v_1^t), \dots, (s_{|QA_t|}^t, v_{|QA_t|}^t)\}$.

4.4 DSG5K: New Diverse DSG Dataset

Existing dialogue datasets rich in information, such as Schema Guided Dialogues (SGD) [62] and MultiWOZ [5], are limited by a small number of domains. Open-domain datasets such as Blended Skills Talk [65] lack the information density for summarizing the slot and values of a dialogue state, rendering them insufficient for building robust DSG models. Meanwhile, previous work has showcased LLM’s proficiency in generating high-quality dialogue data [46]. Thus, the experiments in this chapter test the hypothesis that by creating an information-rich dataset using LLMs across diverse domains, we can build a robust DSG model via symbolic distillation [81]. To this end, a new dataset is generated using LLM dialogue generation named DSG5K, with 5,000+ dialogues spanning 1,000+ domains. Subsequently, `GPTPipe` (Section 4.3) is adapted to annotate DSG5K with silver labels, producing a dataset for training DSG models.

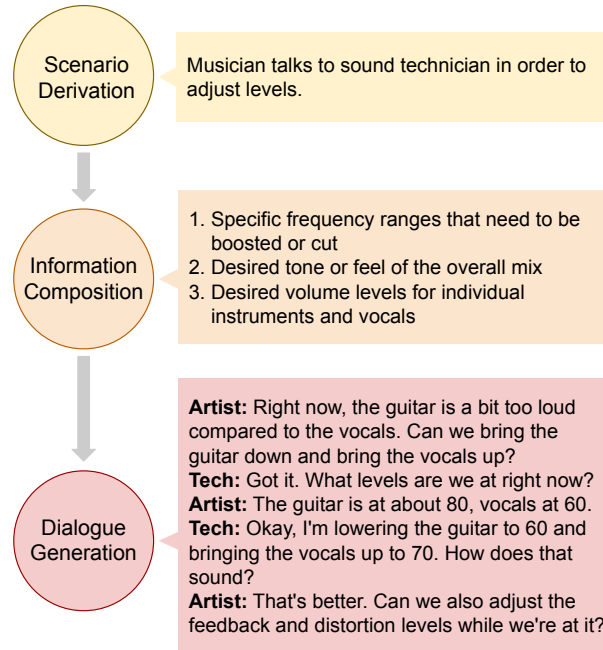


Figure 4.2: Examples of the three stages in the data generation pipeline used to create DSG5K.

Dialogues are generated in three stages. Initially, domains are derived through an iterative process of generating and refining dialogue scenario descriptions (Section 4.4.1). Next, an unstructured list of information types associated with every scenario is generated (Section 4.4.2). Third, a dialogue is crafted based on the scenario description and the unstructured information list (Section 4.4.3). Figure 4.2 gives an overview of this dialogue generation approach.

4.4.1 Scenario Derivation

Algorithm 1 shows the scenario derivation method. GPT is prompted to create a set of k dialogue scenario descriptions (L3). This mini-set is combined with the previously obtained scenarios, where each scenario description is encoded into an embedding by SentenceBERT [63],³ and the resulting embeddings are clustered through a community detection algorithm (L4).⁴ A fresh set of scenario descriptions is created by selecting *any* embedding from every cluster, which is mapped back to its corresponding scenario description (L5). This iteration continues until the set reaches to the

³SentenceBert model: all-MiniLM-L6-v2

⁴https://www.sbert.net/docs/package_reference/util.html

requested size ($L2$). DSG5K was produced using parameters $k = 100, n = 1000$.

Algorithm 1: Scenario Derivation

Input : k : mini-set size, n : final set size.
output : S : the final set containing n scenarios.

```

1  $S \leftarrow \emptyset$ 
2 while  $|S| < n$  do
3    $S' \leftarrow \text{gpt\_generated\_scenarios}(k)$ 
4    $\mathbb{E} \leftarrow \text{cluster}(\text{embed}(S \cup S'))$ 
5    $S \leftarrow \{\forall c \in \mathbb{E}. \text{map}(\text{any}(c)) : c \in C\}$ 
6 end
7 return  $S$ 

```

4.4.2 Information Composition

In a pilot analysis, generating dialogues directly from scenario descriptions (Section 4.4.1) using GPT resulted in generic contents that lack sufficient details for effective DSG model training. To address this issue, an intermediate step was included to compose relevant information for each scenario. In this step, GPT is asked to generate a comprehensive list of information types based on the provided scenario, serving as a knowledge base to retrieve various information associated with the scenario. The GPT output, outlining scenario-oriented information types, is then used to condition the following stage (Section 4.4.3), enhancing the detail and specificity of the generated dialogues.

4.4.3 Dialogue Generation

With a scenario and its associated information types gathered in the previous two stages, GPT is asked to generate a dialogue. The prompt encourages GPT to provide detailed responses and make up values for the information types, while remaining straightforward in its structure.

With this three-stage dialogue generation approach, 1,003 scenarios are derived and 5 unique dialogues are generated per scenario, yielding a total of 5,015 dialogues in the final dataset. Table 4.1 presents the statistics of DSG5K after applying GPTPipe (Section 4.3) to annotate each turn with silver state update labels.

Metric	Value	Metric	Value
Scenarios	1,003	Unique Slots	173,572
Dialogues	5,015	Unique Slots _S	244.6
Turns	100,471	Unique Slots _D	64.9
Turns _D	20.0	Unique Slots _T	3.3
Tokens	2,061,332	Turns w/o SV	1,583
Tokens _T	20.5	Tokens _{SN}	2.4
Slots	324,973	Tokens _{SV}	2.0

Table 4.1: The statistics of the DSG5K dataset (Section 4.4) with dialogue state update labels generated by GPTPipe (Section 4.3). SN/SV: slot names/values respectively, *_{D/T/S/SN/SV}: * per dialogue/turn/scenario/SN/SV, respectively.

4.5 End-to-End (E2E) DSG Model

The DSG5K dataset (Section 4.4) is used to train an end-to-end DSG model, where DSG is framed as a sequence-to-sequence task to predict a dialogue state update U_t for each turn t . A pretrained language model with an encoder-decoder transformer architecture is used as a base model. For training, the dialogue history $D_{*.t}$ and the state update U_t are linearized as described in Figure 4.3. To distinguish speakers and turn boundaries, the speaker tags [S1] and [S2] are prepended to all turns accordingly. For better efficiency, the dialogue history is limited to the previous three turns:

$D_{t-2,t-1,t}$.

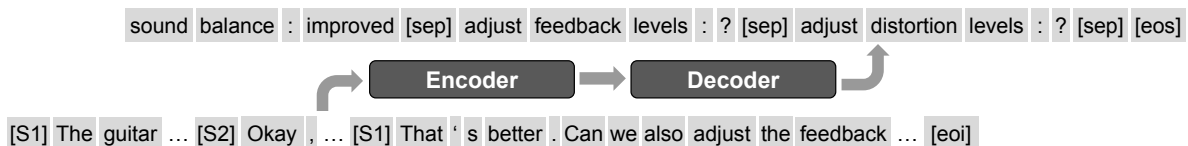


Figure 4.3: Example of input and output formats for end-to-end DSG using a sequence-to-sequence approach.

4.6 Experiments

4.6.1 Datasets

Two datasets are used in the experiments⁵: Schema-Guided Dialogues (SGD), a widely adopted dataset for DST [62], and the new DSG5K dataset (Section 4.4). SGD consists of four held-out domains in its test set, *Alarm*, *Trains*, *Messaging*, and *Payment*, where a held-out domain refers to one not present in the training set. Only the turns related to those four held-out domains (or no domain) compose the evaluation set. Furthermore, to enhance slot diversity, the Schema Guided Dialogues Extension (SGD-X) is included by replacing every slot name in SGD with a randomly selected one from the six alternatives provided in SGD-X for the slot [35]. For DSG5K, one dialogue from each of the 100 held-out domains is selected for the evaluation split, and all turns in those dialogues are included. For each turn t , a dialogue history $D_{t-2,t-1,t}$ is fed into a model, which predicts a set of slot-value pairs for the turn.

	SGD			DSG5K		
	TRN	DEV	TST	TRN	DEV	TST
DO	16	16	4	903	100	100
DI	16,142	2,482	100	4,515	400	100
TU	329,964	48,726	300	90,643	7,816	300
SL	392,383	57,709	416	292,510	25,897	1,012
US	972	701	94	157,834	17,542	852

Table 4.2: The statistics of our data split. TRN/DEV/TST: training/development/evaluation sets, DO: domains, DI: dialogues, TU: turns, SL: slots, US: unique slot names.

Table 4.2 shows the statistics of both datasets. Compared to SGD consisting of 20 domains with 1,284 unique slots, DSG5K demonstrates significantly greater diversity, comprising 1,000 domains with 173,572 unique slot names, despite having a smaller number of dialogues.

⁵SGD with SGD-X slot names was chosen instead of MultiWOZ or vanilla SGD to train our baseline due to superior performance in a pilot study.

4.6.2 Models

Comparison is made among three models for DSG performance, SGD-DSG, E2E-DSG, and GPTPipe. For SGD-DSG, the T5-3B model is fine-tuned [58] using the E2E approach in Section 4.5 on the SGD dataset. For E2E-DSG, a T5-3B-based model is trained using the same E2E approach, but on DSG5K with silver labels from GPTPipe (Section 4.3).⁶ Finally, the zero-shot performance of GPTPipe is assessed in comparison to the other supervised models.

4.6.3 Human Evaluation: Metrics

Due to the generative nature of DSG where both slot names and values must be inferred from the context, evaluating DSG poses a challenge, as traditional DST metrics, such as joint goal accuracy, rely on a fixed slot schema. To facilitate a thorough evaluation of DSG models, a human evaluation measures the following two key aspects:

State Update Completeness measures the proportion of predicted state updates that humans have judged to fully capture the key information in their associated turns. Human judges are asked to read each turn within its context and make a binary decision on whether or not *any* essential information is missing in the state update such that:

$$CP = \frac{1}{|\mathcal{U}|} \sum_{\forall U \in \mathcal{U}} \mathbb{I}(\mathbf{complete}(U))$$

\mathcal{U} is a list of all state updates across dialogues to be evaluated and $\mathbb{I}(x)$ is 1 if x is true; otherwise, 0. Note that the judges are not responsible for finding *all* missing information but identifying at least one to assess completeness for efficient evaluation.

Slot Value Correctness measures the proportion of slot-value pairs that humans have judged to accurately represent *specific* information in their corresponding turns. Judges are asked to mark each

⁶The domains in the training set are mutually exclusive from those in the development and evaluation sets.

slot-value pair as correct if it makes sense and is entirely faithful to the content of the associated turn s.t.:

$$\text{CR} = \frac{1}{\sum_{\forall U \in \mathcal{U}} |U|} \sum_{\forall U \in \mathcal{U}} \sum_{\forall (s,v) \in U} \mathbb{I}(\mathbf{correct}(s, v))$$

Note that both the slot name s and value v must be accurate for $\mathbb{I}(\mathbf{correct}(s, v))$ to be 1.

Three human judges are trained for this evaluation by annotating the state updates predicted by a DSG model on each of 60 turns from the held-out DSG5K data.

Inter-annotator agreement (IAA) is presented in Table 4.5, calculated for 60 turn-predictions annotated by all 3 judges, sampled evenly across both datasets and all 3 models. Correctness IAA is on par with similar human evaluation metrics [15]. Completeness IAA is lower. A manual review of the annotations suggests this is because completeness requires judges to consider more information across an entire state update compared to judging the correctness of a single slot-value pair, leading to higher difficulty.

4.6.4 Human Evaluation: Results

Table 4.3 presents the evaluation results for the three models, `SGD-DSG`, `E2E-DSG`, and `GPTPipe` (Section 4.6.2), on the DSG5K and SGD data (Section 4.6.1) using the *Completeness* (CP) and *Correctness* (CR) metrics (Section 4.6.3). The overall results are estimated by computing the harmonic means (HM) between the CP and CR scores.

Overall, `E2E-DSG` slightly outperforms `GPTPipe` on both datasets, showing the state-of-the-art results. `SGD-DSG` exhibits subpar performance in completeness, scoring only 69.3% in the held-out domains of its own dataset and 32.3% for DSG5K, suggesting that existing DST data, such as SGD, may be insufficiently diverse for developing an effective DSG model. This trend persists in the correctness results; while `SGD-DSG` attains the highest score at 90.8% for SGD, it drops to 72.6% on DSG5K. In fact, the results from `E2E-DSG` and `GPTPipe` are statistically significant compared to those from `SGD-DSG` for all except for CR on SGD by Agresti-Caffo ($p < 0.05$).

`E2E-DSG` achieving around 95% completeness on both datasets holds promising implications

Model	DSG5K			SGD		
	CP	CR	HM	CP	CR	HM
SGD-DSG	32.3	72.6	44.7	69.3	90.8	78.6
E2E-DSG	95.7	81.2	87.9	94.7*	81.7	87.7
GPTPipe	93.3	82.0	87.3	90.0	84.7	87.3

Table 4.3: Human evaluation of SGD-DSG, E2E-DSG, and GPTPipe on DSG5K and SGD for the completeness and correctness. * implies that the E2E-DSG result is statistically significant compared to the GPTPipe result.

for our data generation approach in creating diverse DSG training data and developing general DSG models. While around 81% correctness can be improved, the fact that it does not decrease when evaluated on the out-of-distribution SGD dataset provides further evidence of the robustness of the model.

It is highly encouraging that E2E-DSG achieves competitive performance with GPTPipe, scoring similarly on correctness and outperforming in completeness on both datasets. This implies the possibility of achieving GPT-level performance by fine-tuning a substantially smaller model, T5-3B, with a relatively small dataset. On the other hand, the 82% to 85% correctness achieved by GPTPipe reveals the limitations of the current GPT model in DSG.

4.6.5 Automatic Evaluation: Metrics

Human evaluation is reliable but costly and labor-intensive; thus, the feasibility of automatic evaluation is explored by adapting existing metrics for generative tasks to DSG. However, estimating similarities between predicted and reference state updates as linearized sequences is not suitable for DSG because it fails to account for the slot-value structure. Therefore, an arbitrary similarity metric **sim** is extended to measure the completeness and correctness of a predicted state update against its reference such that it accounts for the structure.

Let \mathcal{U}^p and \mathcal{U}^r be lists of predicted and reference state updates respectively, where $|\mathcal{U}^p| = |\mathcal{U}^r| = n$. Both $U_i^p \in \mathcal{U}^p$ and $U_i^r \in \mathcal{U}^r$ are associated with the same turn in a dialogue. **sim**(x, y) calculates the similarity between x and y . **sim-max**(x, Y) gauges the similarity between x and each item in Y ,

finds $\hat{y} \in Y$ with the highest similarity to x , and returns $\mathbf{sim}(x, \hat{y})$. Let $V_i^* = [s_1^* \oplus v_1^*, \dots, s_k^* \oplus v_k^*]$, where (s_j^*, v_j^*) is the j 'th slot-value pair in U_i^* . Given this formulation, the completeness is calculated as:

$$\frac{1}{n} \sum_{i=1}^n \left[\frac{1}{|V_i^r|} \sum_{\forall v \in V_i^r} \mathbf{sim-max}(v, V_i^p) \right]$$

Similarly, the correctness is calculated as follows:

$$\frac{1}{\sum_{i=1}^n |V_i^p|} \sum_{i=1}^n \sum_{\forall v \in V_i^p} \mathbf{sim-max}(v, V_i^r)$$

For automatic evaluation, various similarity metrics are adapted: BLEU [51],⁷ ROUGE-L [38], BERT score [93], and SBERT similarity [63]. GPT-3.5 is also employed by instructing to match prediction slot-values to gold references (GMatch), assigning a score of 1 when a prediction and reference match and 0 otherwise.

The validity of these automatic metrics is estimated by measuring their agreement with human judgements. This is done by calculating item-level completeness scores for each state update and correctness scores for each slot-value pair that was annotated in the human evaluation (Section 4.6.3). Let \mathcal{H} be a list of binary human judgements, and \mathcal{S} a list of real-valued scores calculated using an automatic metric, such that $|\mathcal{H}| = |\mathcal{S}|$ and each $\mathcal{H}_i \in \mathcal{H}$ and $\mathcal{S}_i \in \mathcal{S}$ correspond to the same item. Given that $A(X, Y)$ calculates agreement between lists of binary judgements X and Y , the agreement between \mathcal{H} and \mathcal{S} is calculated by binarizing \mathcal{S} with a threshold τ that maximizes the agreement:

$$\mathbf{max}(\{A(\mathcal{H}, [\mathbb{I}(\mathcal{S}_i \geq \tau)]_{i=1}^{|\mathcal{S}|}) : \tau \in \mathcal{S} \oplus [\infty]\})$$

This agreement calculation is performed for each automatic metric against majority-vote human judgements, and results are presented in Table 4.5.

⁷SacreBLEU is used with default settings as in Post [54].

	Correctness					Completeness				
	BLEU	ROUGE-L	BERT	SBERT	GMatch	BLEU	ROUGE-L	BERT	SBERT	GMatch
SGD-DSG	12.306	0.177	0.268	0.226	0.868	33.311	0.476	0.764	0.627	0.848
E2E-DSG	5.709	0.099	0.157	0.139	0.231	29.363	0.502	0.720	0.642	0.890
GPTPipe	5.146	0.092	0.159	0.136	0.236	27.200	0.487	0.714	0.636	0.898

Table 4.4: Automatic evaluation results on the SGD evaluation set (Section 4.6.5).

4.6.6 Automatic Evaluation: Results

Automatic evaluation results are presented for the SGD test set only, since gold state references are not available for DSG5K. As shown in Table 4.4, GPTPipe and E2E-DSG approach SGD-DSG’s completeness performance on the held-out SGD evaluation domains, with E2E-DSG slightly outperforming GPTPipe. All models seem to perform poorly on correctness; however, these automatic correctness metrics agree poorly with human judgements as shown in Table 4.5, and are thus unlikely to be reliable estimators of DSG performance. On the other hand, automatic completeness metrics agree with humans about as well as humans agree with each other, suggesting their credibility for evaluation.⁸

	Correctness	Completeness
Human	0.43	0.27
BLEU	0.26	0.17
ROUGE	0.26	0.22
BERT	0.30	0.20
SBERT	0.33	0.28
GMatch	0.17	0.26

Table 4.5: Krippendorff’s Alpha scores for inter-annotator agreement (Human), or agreement of binarized automatic metrics with majority vote human judgments.

⁸Human IAA can be regarded as a soft performance ceiling for agreement between human judges and automatic metrics.

4.7 Discussion

This work introduced DSG as a new task, and demonstrates that it is possible to infer dialogue states in new task domains, even without a slot schema or other in-domain resources. DSG models do not require authoring slot descriptions like zero-shot DSG, and do not require a large number of unlabeled dialogues to discover new slots as with Slot Schema Induction. DSG models are thus the first automatic method for inferring dialogue state in new domains with zero in-domain resource requirements.

The presented experiments demonstrate success of this initial work in tackling the difficult problem of domain generalization in DSG, as the best models show negligible performance drop-off when evaluated on new domains and even out-of-distribution data. Experiment results also show that existing DST resources are currently insufficient for training domain-general models, as the baseline *SGD-DSG* model was unable to adapt to new domains. LLM-based data generation is a promising direction forward that can improve the diversity of training resources at low cost. The results suggest a trade-off where training resources generated via LLMs contain some noise, but are orders of magnitude more diverse and scalable. This enables training DSG models that generalize to virtually any domain, although at times they will make mistakes in their inferences.

One limitation of the work is the simplification of the DSG task formulation, ignoring whether inferred slots conform to a self-consistent schema. As a consequence of this simplification, slot names can be outputted by our DSG models that represent the same information type, but have slightly different surface forms, and this redundancy is not punished by any of the presented evaluation metrics. In practice, consistent slot names are important for linking dialogue states to application infrastructure such as API calls or custom policy logic.

The overall high performance of the *E2E-DSG* model benefits dialogue systems development by providing access to structured dialogue state inferences for virtually any domain with zero cost overhead, apart from running the model. While the correctness performance and consistency of inferred slot names are still important outstanding challenges, this work marks substantial progress towards the goal of inferring dialogue state in a zero-resource setting. The next two chapters build

on this work by leveraging DSG to tackle two tasks. Chapter 5 addresses the schema inconsistency limitation of DSG by investigating DSG’s potential as a value candidate identification approach for Slot Schema Induction. Chapter 6 uses DSG to create a diverse synthetic training resource to improve the domain adaptability of DST models.

Chapter 5

Slot Induction

This chapter describes a novel approach for Slot Schema Induction (SSI). In this task, the goal is to automatically induce critical application-relevant information types as slots from unlabeled dialogue data. As described in Section 2.4, a major limitation of previous work is the use of tagging models to extract slot value candidates directly from dialogue text. Because value candidates must appear in the dialogue text directly, abstractive slots are not modelled properly and might be lost. Furthermore, the value extraction models used were trained primarily for other tasks such as Semantic Role Labeling and masked language modelling, and thus may be poorly optimized for value candidate identification.

The Dialogue State Generation (DSG) approach presented in Chapter 4 serves as an alternative approach to value candidate identification. This approach addresses the limitations of previous value candidate identification models, since DSG directly targets inferring dialogue state information and its identified slot values do not need to appear directly in the text. Also unlike previous approaches, DSG inferences include slot names, which provides an opportunity for automatically providing names for each induced slot cluster.

The presented SSI approach, named DSG-I (DSG-based Induction), thus leverages DSG predictions to obtain slot value candidates, before following a similar strategy as previous works to encode and then cluster candidates into induced slots. Several methods are tried to encode slot value

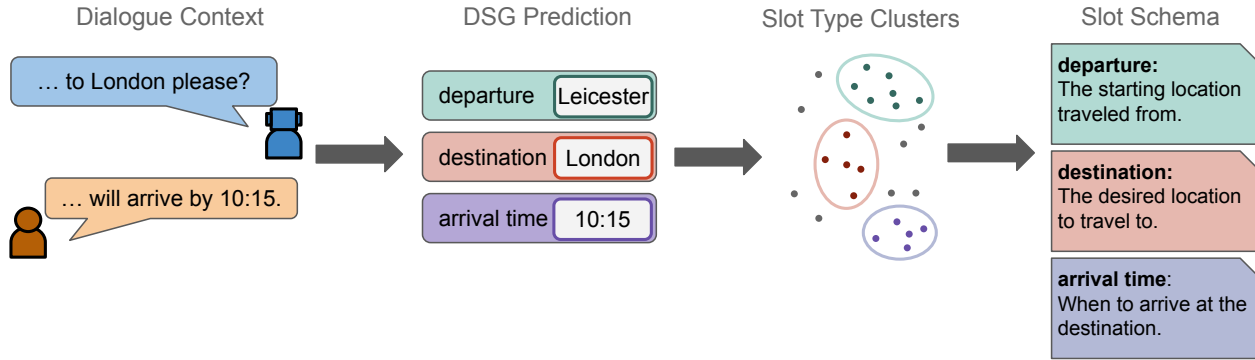


Figure 5.1: DSG-I approach to slot schema induction. The final description generation step is not included in the approach, but is shown for illustrative purposes.

candidates into dense semantic vector representations for clustering. Experiments demonstrate the effectiveness of this DSG-based SSI approach compared to previous work despite its simplicity. Although a limitation of DSG-I is that it does not reliably distinguish slots that encode the same type of information but for different domains, DSG-I produces far fewer redundant clusters than previous works but with similar slot coverage, making it the first practical SSI method.

The presented work thus demonstrates substantial progress on the SSI task, reducing the labor cost of slot schema development by enabling automatic induction of a manageable set of named slot clusters. Induced slots can then be easily reviewed and revised manually to support desired dialogue system behaviors, or used directly to automatically annotate data for DST.

5.1 DSG-I: Inducing Slots from DSG Inferences

Given a set of unlabeled task-oriented dialogues, the presented DSG-I approach produces a list of clusters $\hat{S} = [\hat{s}_1, \hat{s}_2, \dots, \hat{s}_n]$ representing induced slots. Each slot cluster \hat{s}_i is composed of 3-tuples $\hat{s}_i = [(\hat{c}_{i,1}, \hat{n}_{i,1}, \hat{v}_{i,1}), (\hat{c}_{i,2}, \hat{n}_{i,2}, \hat{v}_{i,2}), \dots, (\hat{c}_{i,k}, \hat{n}_{i,k}, \hat{v}_{i,k})]$, where each $(\hat{c}_{i,j}, \hat{n}_{i,j}, \hat{v}_{i,j})$ represents a dialogue context, slot name, and slot value from a DSG prediction. Although the slot names $\hat{n}_{i,1}, \hat{n}_{i,2}, \dots$ within cluster \hat{s}_i may be different, the goal is to produce slot clusters where these names have similar meaning and all values $\hat{v}_{i,1}, \hat{v}_{i,2}, \dots$ of \hat{s}_i belong to the same type of information.

Obtaining slot clusters is performed in three steps. First, a DSG model infers a set of slot-value

pairs for each turn in the unlabeled dialogues, which are combined along with their dialogue contexts into a single candidate list $C = [(\hat{c}_1, \hat{n}_1, \hat{v}_1), (\hat{c}_2, \hat{n}_2, \hat{v}_2), \dots, (\hat{c}_m, \hat{n}_m, \hat{v}_m)]$. Second, an encoding model maps each candidate to a semantic vector representation to obtain $|C|$ vector encodings. Third, a clustering algorithm groups candidates into predicted cluster list \hat{S} by their vector encoding distances.

The DSG-I approach is substantially simpler than previous approaches to SSI, as it relies mainly on the DSG model’s ability to infer a good set of slot value candidates. The main challenge in this approach is to appropriately encode each candidate slot value. Previous work’s encoding of candidate values is straightforward because values are spans from the original dialogue text, and thus contextual embedding models such as BERT can be used to good effect. However, DSG inferred values often do not appear directly in-text. Furthermore, although DSG inferred slot names should provide valuable information for grouping value candidates, it is unclear how to best incorporate them into the encoding. To deal with this challenge, several encoding methods are tried as described in Section 5.1.1. For each encoding method, HDBSCAN is used as the clustering algorithm to obtain final induced slots following previous work [92].

5.1.1 Encoding Slot Value Candidates

In total, six different encoding strategies are experimented with in the evaluation of DSG-I. Each of these methods takes a $(\hat{c}_{i,j}, \hat{n}_{i,j}, \hat{v}_{i,j})$ slot value candidate as input representing a dialogue context, slot name, and slot value corresponding to an inference of the DSG model, and outputs a single vector encoding of the candidate.

The first three encoding methods leverage SBERT as the encoding model, inspired by the results in Section 4.6.6 that suggest SBERT may effectively encode slot-value pairs.

SBERT-S encodes only the slot name.

SBERT-SV encodes the slot name and value using the question mark as a separator.

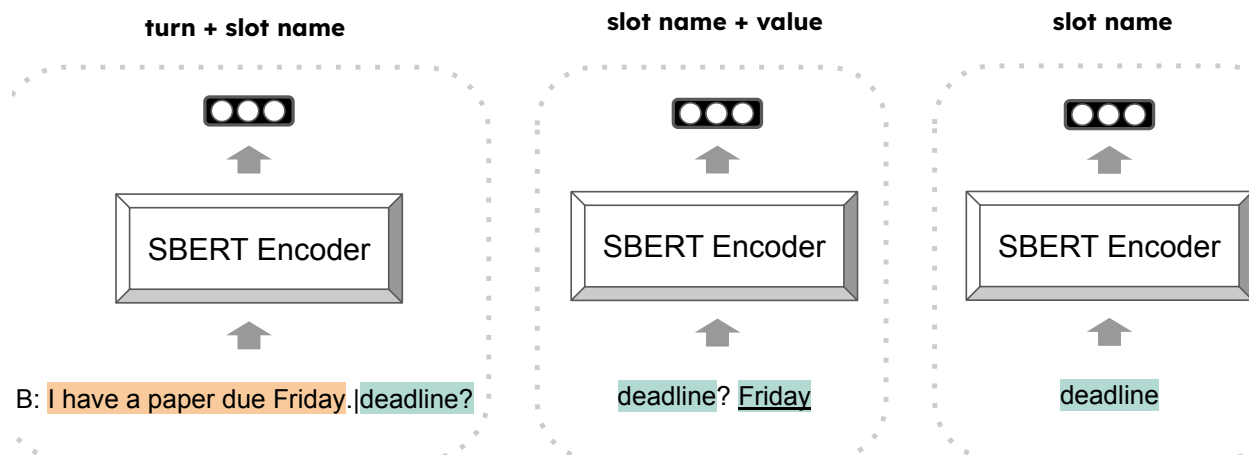


Figure 5.2: SBERT encoding.

SBERT-TS encodes the concatenation of the last turn of the dialogue context only with the slot name. A newline is used as a separator, and a question mark is appended to the end of the slot name in an attempt to give some contextualization to the slot name.

SBERT is a powerful model trained to encode single-sentence text as a fixed width vector, but does not support contextualizing the encoded sentence within a larger body of text. Therefore, the other three encoding methods leverage RoBERTa to encode each token in the slot value candidate with the dialogue context as a single sequence, before averaging the encodings of a relevant token subsequence to produce a final single vector representation for the candidate. For all three methods, the candidate token sequence is formed by concatenating the full dialogue context $\hat{c}_{i,j}$ with the slot name $\hat{n}_{i,j}$ and value $\hat{v}_{i,j}$. To attempt to suggest the semantic relationship between the components of this concatenated sequence, a newline separator is used between the context and slot name, and a question mark separator is used between the slot and value.

ROBERTA-S averages the contextualized token embeddings of the slot name only.

ROBERTA-SV averages the contextualized token embeddings of the slot name, question mark separator, and value.

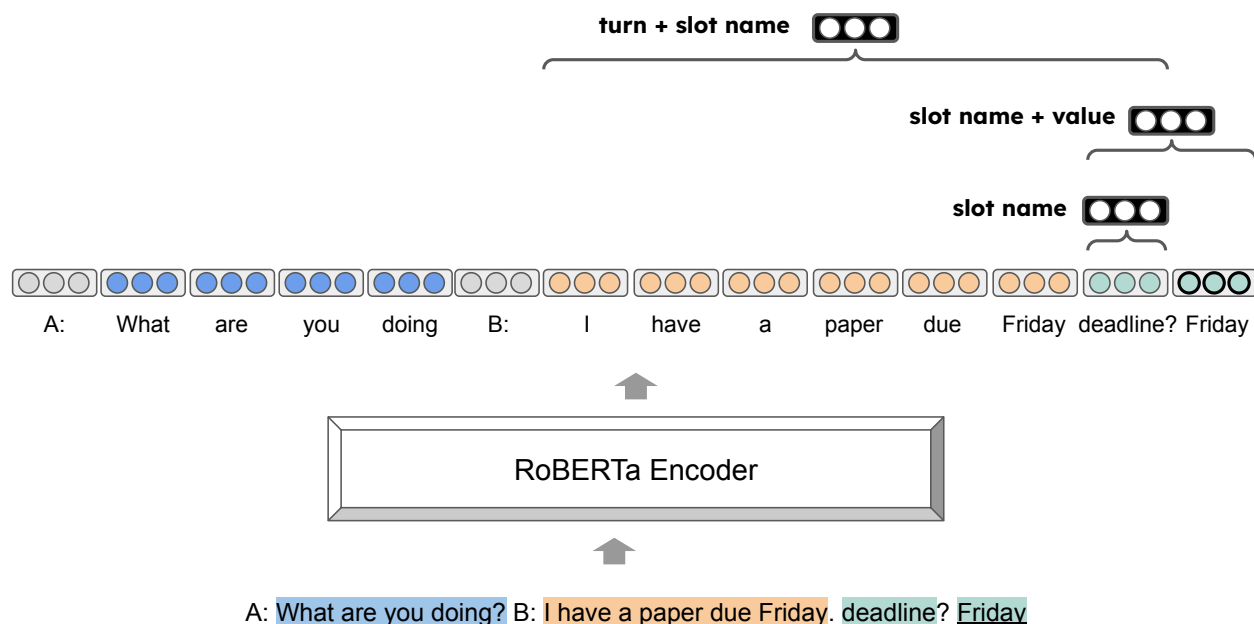


Figure 5.3: RoBERTa encoding.

ROBERTA-TS averages the contextualized token embeddings of the final turn of the dialogue context, newline separator, and slot name.

5.2 Experiments

5.2.1 Data

To facilitate comparison with previous work, the validation split of the MultiWOZ 2.1 dataset is used for evaluating the DSG-I approach. The MultiWOZ slot schema is used as a gold reference, including only the main 5 domains: Taxi, Train, Restaurant, Attraction, and Hotel. Since there is a large amount of overlap in the slot specifications across domains as seen in Table 5.1, the evaluation is performed in both *Domain-Aware* (DA) and *Domain-Free* (DF) settings.

Domain-Aware All 31 MultiWOZ slots are considered to be unique slot types. In this setting, the SSI model is expected to induce separate slots for different domains, even if the semantic meaning

of the slot is the same. For example, separate `Leave at` slots should be induced distinguishing the time for a taxi departure versus the time for a train departure.

Category	Taxi	Train	Restaurant	Attraction	Hotel
Leave at	✓	✓			
Arrive by	✓	✓			
Departure	✓	✓			
Destination	✓	✓			
Day		✓			
Book People		✓	✓		✓
Book Day			✓		✓
Book Time			✓		
Food			✓		
Name			✓	✓	✓
Price Range			✓	✓	
Stars			✓	✓	✓
Area			✓	✓	✓
Type					✓
Book Stay					✓
Internet					✓
Parking					✓

Table 5.1: Slots present in each of the main 5 MultiWOZ domains.

Domain-Free The 31 MultiWOZ slot types are consolidated into the 17 Domain-Free slot types seen in Table 5.1 based on shared name, since slots sharing a name carry the same semantic meaning. This setting is a better evaluation of the SSI approach’s ability to induce a robust set of unique information types, but domain information is lost.

5.2.2 Evaluation Metrics

For comparability, the evaluation of DSG-I follows the formulation and metrics laid out in previous work. Given an induced list of slot clusters $\hat{S} = [\hat{s}_1, \hat{s}_2, \dots, \hat{s}_n]$ where each slot \hat{s}_i is a cluster of values $\hat{s}_i = [\hat{v}_1, \hat{v}_2, \dots, \hat{v}_{|\hat{s}_i|}]$, and a list of gold reference slots $S = [s_1, s_2, \dots, s_m]$, the evaluation aims to calculate the quality of matching induced slots to gold references. To do this, a mapping

$M : \hat{S} \rightarrow S \oplus [\text{none}]$ is constructed by hand¹ to assign each predicted cluster to a gold slot (or to none if no gold slot matches).

Manual slot mapping for evaluation is performed by a graduate student by viewing a random sample of 10 slot-value pairs produced by the DSG model, along with their dialogue contexts, for each induced cluster, and assigning the cluster to a gold slot only if the vast majority (> 7) of slot-value pairs accurately reflect the semantics of some gold slot; otherwise, the predicted slot is mapped to `none` and treated as a precision error. Additionally, because the gold MultiWOZ slot schema does not have full coverage of all types of task-critical information introduced in its dialogues, predicted slots were excluded from the evaluation entirely if they reflect a valid and task-relevant information type but there is no corresponding gold slot. This was done primarily for information types typically introduced by the agent speaker such as phone numbers, reference numbers, postcodes, and vehicle types. The MultiWOZ slot schema does not attempt to cover these information types in the slot schema because their values are meant to originate from the hypothetical dialogue system, thus there is no need to track them as slots.

Given the mapping M from predicted to gold slots, the evaluation metrics are calculated follows:

Slot Precision is the proportion of predicted slots that were able to be mapped to a gold slot:

$$\text{SP} = \frac{\sum_{\hat{s}_i \in \hat{S}} 1_S(M(\hat{s}_i))}{|\hat{S}|} \quad (5.1)$$

Slot Recall is the proportion of gold slots for which there is at least one corresponding predicted slot:

$$\text{SR} = \frac{|\{M(\hat{s}_i) : \hat{s}_i \in \hat{S}\} - \{\text{none}\}|}{|S|} \quad (5.2)$$

¹One previous work uses an automatic matching method based on cosine similarity of vector encodings, but this evaluation method was found to be easily exploited to overestimate SSI performance in a pilot study. Therefore, the presented work adopts the method of manually mapping predicted to gold slots.

Slot F1 is calculated normally as the harmonic mean of precision and recall:

$$\text{S-F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (5.3)$$

Number of Induced Slots In the above Slot Precision calculation, multiple predicted clusters are allowed to be mapped to a single gold slot. This choice of formulation was made by previous work to avoid punishing the schema induction approach for inducing a finer-grained schema than what the gold schema provides, but fails to reflect the number of redundant clusters that are induced. To mitigate this, the number of induced slots is reported as an additional evaluation metric, where a lower number of induced slots is considered preferable.

Value Precision is meant to measure the purity of predicted slot clusters. It is calculated only between matched predicted clusters $\hat{S}_{matched}$ and matched gold clusters $S_{matched}$. For each gold slot with at least one match $s_i \in S_{matched}$, the proportion of predicted values in the mapped predicted slots that have a fuzzy match to some gold slot value is measured using fuzzy match boolean function f :

$$\text{VP}_{s_i} = \frac{|\{\hat{v}_{kl} : \hat{v}_{kl} \in \hat{v}_k, M(\hat{s}_k) = s_i, v_{ij} \in s_i, f(v_{ij}, \hat{v}_{kl})\}|}{|\{\hat{v}_{kl} : \hat{v}_{kl} \in \hat{v}_k, M(\hat{s}_k) = s_i\}|} \quad (5.4)$$

The final Value Precision score is an average across matched gold slots calculated in this way:

$$\text{VP} = \frac{\sum_{s_i \in S_{matched}} \text{VP}_{s_i}}{|S_{matched}|} \quad (5.5)$$

Value Recall is calculated similarly to Value Precision. For each gold slot with a mapping to one or more predicted clusters, recall is measured as the proportion of gold values that have a fuzzy match to some value in the corresponding predicted clusters:

$$\text{VR}_{s_i} = \frac{|\{v_{ij} : \hat{v}_{kl} \in \hat{v}_k, M(\hat{s}_k) = s_i, v_{ij} \in s_i, f(v_{ij}, \hat{v}_{kl})\}|}{|s_i|} \quad (5.6)$$

The final Value Recall is also averaged across matched gold slots:

$$\text{VR} = \frac{\sum_{s_i \in S_{\text{matched}}} \text{VR}_{s_i}}{|S_{\text{matched}}|} \quad (5.7)$$

Since previous work does not specify the fuzzy matching algorithm used, fuzzy matching is performed using the Python standard library implementation `difflib.get_close_matches` of the Ratcliff and Obershelp algorithm, with a default threshold of 0.6.

5.2.3 Pilot Evaluation

To avoid a large evaluation cost incurred by the manual mapping of predicted slot clusters to gold slots, a pilot was conducted using a heuristic automatic evaluation to select an encoding strategy. The automatic evaluation constructs the mapping M by assigning each predicted cluster to the gold slot s_i in a way that maximizes the Value Recall VR_{s_i} (Eq. 5.6) between the cluster and the gold slot. If no assignment can be found that results in a Value Recall above a 20% threshold, the predicted cluster is assigned to `none`. This automatic evaluation method is somewhat noisy, but enables efficient filtering of poor encoding methods.

Table 5.2 presents the results of the pilot evaluation, in which each of the 6 encoding methods presented in Section 5.1.1 was evaluated using the heuristic evaluation method 30 times with a random grid search of the hyperparameter space², and the top result for each encoding method is compared. To mitigate the curse of dimensionality, the UMAP algorithm was used for dimensionality reduction in 15 of the 30 evaluation runs for each encoding strategy.

Pilot results indicate the superior performance of the SBERT-S and SBERT-SV encoding methods, with UMAP dimensionality reduction appearing to hurt performance for these two strategies. A brief review of each approach’s clustered outputs revealed that, although SBERT-S showed a slightly higher performance on Slot F1 in the pilot, the clusters it produces often conflate

²Although the encoder models require no hyperparameters to be set for encoding, HDBSCAN and UMAP each have several hyperparameters.

Model	Encoding	UMAP Used	Slot Precision	Slot Recall	Slot F1
SBERT	s	No	42.9	93.5	58.8
SBERT	sv	No	45.3	74.2	56.2
RoBERTa	sv	Yes	35.0	54.8	42.7
RoBERTa	s	No	78.6	25.8	38.9
RoBERTa	ts	Yes	28.6	29.0	28.8
SBERT	ts	Yes	26.3	25.8	26.1

Table 5.2: Pilot Results

several distinct types of information. Table 5.3 illustrates a typical example of this problem, where a "guest house" cluster is induced due to the prevalence of this substring in the candidate slot name produced by DSG, yet the values are not homogeneously typed.

Slot name	Slot value
expensive guest house	No
moderate priced guest houses	3
guest house type	Expensive
moderate guest houses east side	“carolina bed and breakfast”, “warkworth house”
moderate priced guest house	hobsons house
guest houses meeting criteria	12
guest houses location	East
guest house help	True
guest house 1	a and b guest house
guest house west side	2

Table 5.3: Example of 10 slot value inferences from the DSG model, sampled from a single cluster produced by DSG-I using the SBERT-S encoding strategy.

5.2.4 Models

DSI [47] leverages a Part-of-Speech (POS) tagger, Named Entity Recognition (NER) tagger, and coreference resolution model to extract value candidate spans using a set of heuristic rules. Slot clusters are then assigned to value candidates using a neural latent variable model.

USI [92] is the SoTA approach. It is a fully unsupervised SSI approach that leverages attention scores between token spans estimated using a pretrained language model to extract value candidates.

A three-step hierarchical clustering procedure is then used that aims to cluster value types, then domains, then slots, using HDBSCAN as the base clustering algorithm.

DSG-I is the presented approach. In light of the pilot evaluation results, SBERT-SV was chosen as the encoding strategy. Since DSG-I relies on HDBSCAN clustering for which hyperparameters must be tuned to achieve good performance on a particular dataset, a single graduate student tuned `min samples` and `min cluster size`³ by iteratively running the DSG-I approach on the test set⁴ and manually reviewing a random sample of slot values in each predicted slot cluster. Although ideally such manual tuning would be unnecessary, this process was straightforward due to the relative insensitivity of the approach to different hyperparameter values and manageable number of induced clusters; the entire hyperparameter tuning process took less than 10 minutes and resulted in the values `min samples = 75` and `min cluster size = 30`.

5.2.5 Results

Setting	Method	Slot Type			Slot Value			
		# Clusters	Precision	Recall	F1	Precision	Recall	F1
DA	DSI*	522	96.15	80.65	87.72	41.53	57.40	37.18
	USI*	290	100.00	93.55	96.67	61.34	67.26	58.71
	DSG-I	69	93.75	70.97	82.23	79.07	73.12	72.20
DF	DSG-I	69	94.44	100.00	97.14	77.59	68.06	64.44

Table 5.4: Schema induction results on the MultiWOZ 2.1. DA: Domain-aware evaluation setting. DF: Domain-free evaluation setting. *DSI and USI results are taken from Yu et al. [92].

Evaluation results are presented in Table 5.4. DSG-I produces far fewer clusters than previous work, indicating far fewer redundancies in induced slots. The precision of induced slots is comparable to previous approaches; in fact, only 2 clusters were produced did not correspond to any

³Maximum cluster size was left unconstrained.

⁴This makes a fair comparison to baseline approaches, as the validation split of MultiWOZ was used for both development and testing in previous work.

well-defined and task-related information type⁵. DSG-I has inferior coverage of Domain-Aware slot types compared to previous approaches, as indicated by a lower Slot Recall. The reason for this is that it does not reliably distinguish domain boundaries when the type of information represented by a slot is the same across domains. This is demonstrated by a 100% recall score in the Domain Free evaluation setting.

Induced DSG-I slot clusters have higher purity and coverage of gold values for the corresponding slot. This is indicated by a higher Value Precision and Value Recall of DSG-I. Note that the Value Recall of DSG-I drops in the Domain-Free evaluation setting because the model is expected to cover values for additional domains for which it did not output a separate slot cluster in the Domain-Aware setting. For example, DSG-I does not induce a `Taxi Arrive By` cluster because the DSG model’s naming of this slot for the Taxi domain is somewhat noisy. However, DSG-I does induce a `Train Arrive By` cluster. In the Domain-Free setting, these slots are treated as a single slot induction target, leading to many value misses on Taxi examples when measuring the Value Recall of `Arrive By`.

5.2.6 Analysis

To illustrate the induction abilities of DSG-I, Table 5.5 provides 3 randomly sampled examples of every induced slot cluster that was mapped to a gold Hotel slot in the Domain-Aware evaluation. Although the examples overall appear to faithfully induce MultiWOZ slots for this domain, it can also be seen from these examples that the SBERT-SV encoder can be sensitive to the surface form of the slot name. This causes separate clusters to be induced that are semantically equivalent, such as the separate `num nights` and `stay duration` clusters that both represent the `hotel book stay` slot.

Analysis of Missed Slots An error analysis of all Domain-Aware gold slots that were not induced by DSG-I was conducted by randomly sampling 10 gold slot-value examples for each of the 9 missed

⁵37 clusters were excluded from the evaluation, as they clearly represented other task-related information types not covered by the gold slot schema. A breakdown of these excluded clusters is presented in Section 5.2.6.

Book Stay	
A: thank you . i would also like entertainment options in the same area as you booked the hotel .	num nights: 3
A: yes , i need it for 2 people and 5 nights starting from tuesday . i would also like the reference number .	num nights: 5
A: can you book it for 4 people for 4 nights starting on thursday ?	num nights: 4

A: yes , i would like to book 5 nights at the leverton house .	stay duration: 5 nights
A: book it for thursday night , 3 nights , and 4 people . thank you !	duration: 3 nights
A: it s just 1 night for 4 people .	stay duration: 1 night
Internet	
A: i do not have an area preference but it needs to have free wifi and parking at a moderate price .	free wifi: Yes
B: hobsons house is a moderate -ly priced , 3 star guest house in the west . it offers free internet and parking . would you like me to book this for you ?	free internet: yes
A: great , can i please have the address and phone number ?	
B: the cambridge belfry also include -s free wifi . i would be more than happy to make your reservations , would you like to do so ?	wifi situation: Free
A: yes make reservations for 6 people for 5 nights	
Parking	
A: please pick 1 that has free parking .	free parking: 1
A: i would prefer something with 2 stars , and i need free parking as well .	parking requirement: free parking
A: i am looking for someplace to stay . i would like something expensive , and i do not need parking .	parking needed: No parking needed
Stars	
A: i am flexible about cost , i would prefer free wifi and a 4 star rating though .	4 star rating: Yes
B: the ashley hotel and the lovell lodge are both in the north and are in the moderate price range with 2 stars .	star rating: 2 stars
A: will you book it for a party of 7 . 5 days beginning friday .	
A: the star preference does not matter . i'll take the 4 star .	star preference: 4

A: yes , i am also looking for a 3 star hotel located in the same area as the restaurant .	hotel type: 3 star
A: the hotel should include free parking and should have a star of 3 .	hotel star rating: 3
A: i am looking for a 4 star hotel in the south .	hotel rating: 4 star
Book Day	
A: yes , please recommend 1 that can accommodate 8 people for 2 nights starting on thursday	start date: Thursday
A: 4 people for 2 nights starting sunday , please !	start date: Sunday
A: great ! i would like to book rooms for 5 people for 4 nights starting friday .	start date: friday
Area	
A: hi , i am looking for a moderate -ly priced hotel in the east .	hotel location: East
A: i am looking for a hotel with free parking in the north .	hotel location: North
A: i need a hotel in the south with free wifi .	hotel location: South
Price Range	
A: thank you , i am also looking for a hotel in the east that has free parking for my car . i'll need the hotel to be cheap with wifi	hotel budget: Cheap
A: hi , i am looking for a moderate -ly priced hotel in the east .	hotel type: Moderate -ly priced
A: hmm , well how about checking for a hotel instead that is expensive with free parking and wifi ?	hotel type: Expensive
Type	
A: thanks i am also needing to book at a guest house for 6 for 5 nights starting on sunday	room type: Guest house
A: let s go with a hotel . i would actually like it to have free internet and parking , if possible .	hotel or guest house: Hotel
A: i am looking for a place to stay . ideally , it will be a guest house and have a 4 star rating .	accommodation type: Guest house

Table 5.5: Examples of DSG-I results mapping to the gold Hotel slots of MultiWOZ. Dashed lines separate individual DSG-I clusters that map to the same gold slot.

slots: taxi departure, taxi arrive by, taxi destination, taxi leave at, attraction area, attraction name, hotel name, hotel book people, restaurant area. These 90 slot-value examples were manually reviewed in their original dialogue contexts, alongside the DSG model prediction for those contexts to infer a cause of each error. The results of this analysis are presented in Table 5.6.

For 17 cases, no issue with the DSG predictions could be determined and the inferred slot values appeared to be correctly clusterable by DSG-I, indicating that the density of cases similar to these was not high enough to capture the corresponding gold cluster. The most prevalent issue with the DSG model predictions was a lack of domain information occurring in 25 cases, such as `departure time: 13:30` being inferred. These slot value inferences were frequently clustered into another domain's slot cluster of the same name, explaining the difference in Slot Recall performance seen in the evaluation. The next most prevalent error category was termed "Slot Variant," which we use to denote the 23 cases where the slot name produced by the DSG model was strange in some way, though not technically incorrect. For example, the slot name `arrival restaurant` contains the relevant semantic information corresponding to the intended gold `restaurant destination` slot, but the naming quality is poor. This noisy naming reduced the density of slot value encodings for the `hotel name` and `attraction name` slots especially, resulting in no cluster forming for these slots. Only in 5 out of 90 cases did the DSG model simply miss the relevant slot and value entirely. Other errors were due to failures in semantic understanding by the DSG model, such as mixing up the semantics of destination vs. departure or including the value in the slot name.

5.3 Discussion

DSG-I demonstrates several improvements on the SSI task, notably a vast reduction of redundant induced slots, as well as better cluster purity and coverage of gold values. This result capitalizes on the ability of the DSG approach presented in Chapter 4 to infer a high-coverage set of slot values

across unlabeled dialogue data and is further evidence for its efficacy. However, the evaluation also reveals that DSG-I is limited in its ability to distinguish domains, as evidenced by a lower slot recall on the Domain Aware evaluation compared to previous methods. This limitation can likely be mitigated by incorporating a dedicated domain induction component as was used in previous work [92], and this direction should be explored in future work. In many realistic settings however, the application setting may be constrained to a single domain, or the distinction of domains for the same slot type may not be important, in which case the evaluation suggests DSG-I is highly practical. DSG-I thus marks a step forward for SSI, reducing the labor cost of manual slot schema creation and consequently increasing the viability of integrating conversational AI in real-world applications.

Error	Definition	Example	#
No Domain	The predicted slot name lacks specification regarding the domain to which the information pertains.	A: would you be able to arrange taxi service from cambridge artworks to royal standard leaving by 13:30 ? departure time: 13:30	25
Slot Variant	The predicted slot name is irregular compared to other predicted slots conveying similar types of information.	A: i would like to leave the hotel by 22:00 to go to the gardina . please let me know the contact number and car type . arrival restaurant: gardina	23
Correct	The predicted slot name and value accurately represent the information conveyed in the dialogue turn.	A: i need a taxi between the hotel and the camboats . i want to leave the camboats by 3:45 . taxi departure time: 3:45	17
Coreference	The predicted value is the reference term, instead of being resolved to the appropriate entity mentioned in a previous turn.	A: what types of college are they ? B: may i recommend hughes hall . it is in the centre of town and has free entrance ? ... A: i also need a taxi to get from college to the restaurant . i need to arrive by 14:45 . destination: College	8
Value in Slot	The slot value leaks into the slot name.	B: the lensfield hotel has both free parking and internet it s in the south , would you like me to book it for you ? A: yes , please book the room . book lensfield hotel: True	7
Missing	The system fails to predict a specific slot and its corresponding value in the dialogue turn.	A: i heard of that place from a friend , not sure if i like that 1 . . but it would be for just me for 2 nights starting sunday . num nights: 2 (correct) start date: Sunday (correct) book people: 1 (missing)	5
Destination/Departure	The assignment of departure and destination properties is inverted, resulting in the departure details being mistaken for destination information and vice versa.	A: i want to be at the attraction by 2:15 and will need a contact number for the driver and car type . leave time: 2:15	3
Incorrect Value	The predicted slot value is inappropriate for the specified information type, resulting in an erroneous slot name and value pair.	A: i'll need the taxi between charlie chan and castle galleries please taxi destination: charlie chan, castle galleries	2

Table 5.6: Error analysis

Chapter 6

Domain-Adaptable Dialogue State Tracking

This chapter presents a novel approach for zero- and few-shot dialogue state tracking (DST), and aims to answer the research question: **to what extent does increasing the diversity of dialogue state tracking data improve the domain adaptability of trained DST models?** As discussed in Chapter 2, a major limitation of existing work in DST is the low diversity of training data. The two most popular datasets, MultiWOZ and SGD, only cover 5 and 16 domains respectively, with only 31 and 214 slot types respectively. This limited diversity may inhibit their effectiveness as training data for real-world applications, since the DST model will need be generalizable enough to robustly adapt to the unique slots required by any particular application domain.

The DSG approach presented in Chapter 4 allows investigating the impact of slot diversity on DST domain adaptation performance on an unprecedented scale, since it provides access to automatically-annotated dialogue state labels for virtually any domain. This Chapter therefore presents experiments using a DSG-based data generation approach as a pretraining step for zero- and few-shot DST. Results reveal a substantial performance boost of this novel approach over strong baselines, demonstrating that the diversity of conventional training resources is indeed a bottleneck for domain adaptability on this task.

6.1 Related Work

Several previous works explore data augmentation methods for improving the quality and diversity of limited DST data. Nearly all of these approaches target the low-resource or few-shot settings, using data augmentation techniques to improve the effectiveness of limited in-domain training resources. This can be done using simple approaches to improve the lexical or semantic diversity of training examples, or by synthesizing entire dialogues to create additional training resources.

Lexical Diversification One approach is to use paraphrasing techniques to improve lexical diversity on the turn-level. Quan and Xiong [57] experiment in this direction with a variety of methods such as back-translation and synonym replacement, and Yin et al. [91] use a reinforcement learning approach to learn to replace token spans with paraphrases. These works demonstrate the potential of data augmentation to improve existing training resources, but their focus on paraphrasing fundamentally limits the extent to which the original data can be altered since the goal is to maintain the semantic content of original examples.

Semantic Diversification Other approaches look to improve the generalizability of trained DST models to handle new values and dialogue contexts by modifying the semantic content of original dialogues. Summerville et al. [72] focus specifically on the problem of DST models' ability to generalize to new slot values, using external corpora to augment training data with additional values for open-ended slot types. Lai et al. [33] synthesize new training examples by generating a new response to the context of existing dialogues. Their response generator is conditioned on the dialogue act and state, but is given a new dialogue act and state during augmentation to increase the semantic diversity of the training pool. These works successfully augment the lexical and semantic content of DST training data on the turn- or slot-value-level.

Dialogue Reconstruction Some works augment existing data by synthesizing entirely new dialogues from an initial seed set. Three works explore methods that take advantage of the state representations in DST data to create a state transition graph, and then generate entirely new

dialogues by traversing transition paths that are not represented in the initial dataset [1, 2, 6]. Once a new state transition path for a synthetic dialogue is sampled from the transition graph, the turns from the original dialogues corresponding to each transition are used as templates and filled with new slot values to produce a final natural language dialogue. This approach introduces new variations in the structure and content of training data. However, the synthetic dialogues produced will share many of the same features as the original seed data, especially due to the reliance on templates. Mehri et al. [44] use a similar approach but eliminate the reliance on seed dialogues by using slot schema specification to create the state transition graph, and GPT-3 is used to paraphrase each template-generated turn to be more natural and coherent. It is difficult to evaluate the efficacy of their method however, since less-common evaluation data MixSNIPS/MixATIS [55] are used making comparison to related work difficult.

Full Dialogue Generation Three recent works generate new DST data by training PLMs to generate new dialogues from a task goal and schema definition. Kim et al. [29] trained a dialogue generator model to produce dialogues given a goal, schema, and queryable database of schema values, and trained separate dialogue state labeler model to label the generated dialogues with dialogue states. Mohapatra et al. [48] train a pipeline of separate PLMs to model a user response generator, user response selector, dialogue state generator, system response generator, and system responses selector. Wan et al. [77] similarly trained separate PLMs for to simulate user and system agents. They demonstrated improved transfer to generating synthetic data on low-resource target domains by pre-training their simulation agents on 12 different training data from previous work. All three of these approaches target low-resource DST by training their dialogue generation models on a limited amount of in-domain data, then train the DST model on synthetically generated data. Their results demonstrate the power of using PLMs to generate data to domains where substantial training resources are unavailable.

The works highlighted above demonstrate that automatic methods for data augmentation and

generation can help address the limitations of existing training resources and improve transfer to data-poor domains. One area in this direction for which there is little work is generating data for new domains entirely from scratch or with a minimal schema specification. The recent PLM-based work [29, 48, 77] for low-resource DST is the closest to this goal, but these works still rely on a small amount of in-domain training data or other domain-specific resources. Unlike any previous data generation or augmentation approach in DST, the presented approach based on DSG synthesizes entirely new domains, slot types, and full dialogues from scratch and can be run fully automatically with no human intervention.

6.2 Approach

The presented DST approach consists of three stages. First, an extension of the fully automatic data generation approach used to generate DSG5K from Chapter 4 is used to create a synthetic DST dataset. Second, an end-to-end model is pretrained on the synthetic data to acquire domain-general DST ability. Third, because synthetic data can be noisy, a fine-tuning stage trains the model similarly to conventional approaches using gold DST annotations to further refine the model.

Synthetic Data Generation To generate a synthetic DST dataset from scratch, a set of dialogues covering diverse task scenarios is first generated using the approach from Section 4.4, then each turn in the dialogue is annotated with a dialogue state update label using a DSG model as in Section 4.6. This data generation method alone yields data that supports training a zero-shot DST model by predicting values given slot names within the context of each dialogue. However, previous work has shown the value of using in-context learning for zero- and few-shot DST using natural language descriptions and few shot examples. Therefore, we extend the data generation pipeline using LLMs to generate a description and list of value examples for each slot value produced by the DSG model.

Given each dialogue context, slot, and value update tuple in the generated data (c_i, s_i, v_i) , an instruction-tuned LLM generates a one-sentence natural language description d_i for the slot s_i using a prompt that includes all information in the tuple. Similarly, a LLM generates a comma-separated

list of alternative value examples e_i given a prompt that includes all information in (c_i, s_i, v_i, d_i) , in order to further define the slot. Note that these value examples do not qualify as few shots because they are not grounded in any dialogue context, and primarily serve to illustrate the valid space of values that can fill each s_i . The resulting list of each 5-tuple $(c_i, s_i, v_i, d_i, e_i)$ defines the full set of information in the synthetic DST dataset. These examples can be used to train an end-to-end, sequence-to-sequence dialogue model such as T5 using the sequence format shown in Table 6.1.

A: Good afternoon, Mr. Smith. I'm here today to survey your land and assess its value. May I ask you a few questions?

B: Of course, please go ahead.

A: Firstly, can you tell me the location and size of the land?

B: Sure. The land is located on the outskirts of town, about 10 miles away from the city center. It's approximately 20 acres in size.

A: That's helpful. Can you also tell me about the type of terrain and land features on the property?

Identify the information from the above dialogue:

land size : the area encompassed by the property, typically measured in units such as acres, hectares, or square miles.

ex. The floodwaters have submerged over 150 hectares of farmland. land size? -> 150 hectares

ex. Yes, we're finalizing a purchase of 50 acres in the valley. land size? -> 50 acres

ex. I am going to the wildlife sanctuary. It spans 80 square miles. land size? -> 80 square miles

(e.g. 50 hectares, 2 square miles)?

Figure 6.1: An example of an input token sequence format used for training. The dialogue context c_i (yellow) and slot name s_i (peach) is taken from the synthetic DSG5K data, and extended with an automatically generated slot description d_i (green) and value examples e_i (red). Few shots f_i (purple) are optionally used in the few-shot setting and collected automatically using the DSG-I slot induction approach presented in Chapter 5. The value example $v_i = 20$ acres is the silver output sequence for this training example.

Empty Slot Sampling One challenge of using DSG-generated labels for DST training is that DSG never produces examples of empty slots, since new slots are inferred only to capture observed values. Naively training a DST model to predict each DSG inferred value v_i given (c_i, s_i, d_i, e_i) will cause the model to hallucinate non-empty slot values during test time. On the other hand, constructing training data that includes an example for every combination of slot name and dialogue context

from the same domain overrepresents empty slots because DSG produces such a wide variety of slot names, causing the model to be too conservative in filling slots with values. Therefore, a sampling strategy is adopted where, given n slot-value examples inferred by the DSG model, αn empty slot examples are added to the data. Each empty slot example is constructed by replacing the slot definition (s_i, d_i, e_i) of some randomly-selected training example $(c_i, s_i, v_i, d_i, e_i)$ with (s_j, d_j, e_j) from the j th example in the original dataset, in conjunction with replacing v_i with the empty value n/a . j is sampled from a uniform distribution across all examples in the original dataset, as validation experiments revealed this method to be superior compared to restricting the sampling to the within the same domain as the original example i .

Few Shot Sampling Constructing few-shot training examples is another challenge when using DSG for few-shot DST data generation. Conventionally, training shots can simply be sampled from other examples containing the slot from within the same training data. However, DSG slots do not conform to a consistent schema, making this strategy nontrivial. To overcome this, the DSG-I slot schema induction approach presented in Chapter 5 is used to cluster slot value examples. Then, for each training example $(c_i, s_i, v_i, d_i, e_i)$, a set of up to k few shots $f_i = (t_1, v_1), (t_2, v_2), \dots, (t_k, v_k)$ is sampled from within the same cluster, where each t_j is the last turn of dialogue context c_j and each v_j is a value that was inferred to match the definition of slot s_i for that turn. Few shot sampling is constrained to shots that come from a different dialogue but within the same domain. These constraints were chosen based on manual review of sampled few shots during validation experiments, in which it was discovered that the values of shots sampled from the same dialogue or from different domains as target example i often did not match the original description d_i .

Simulated State Tracking Conventionally, DST models track slots as they are updated throughout the course of a dialogue. However, the DSG approach from Chapter 4 infers only slot-value pairs that are expressed locally for each dialogue turn. To align the synthetic training resource better with the DST task, examples are altered to train the model to infer values from the entire dialogue context, rather than from the final turn of the context. To do this, the context c_i of each training

example $(c_i, s_i, v_i, d_i, e_i)$ is replaced with some c_j from the dialogue continuation. Specifically, if c_i ends on turn k and the complete dialogue has n turns, c_j is sampled uniformly from contexts ending within turn indices $[k, n)$.

6.3 Data

Experiments are presented on the MultiWOZ dataset, as it is the most popular data for DST. MultiWOZ 2.4 is used as it contains corrected gold labels in the validation and test splits. An evaluation on MultiWOZ 2.1 is also included to facilitate further comparison to previous work.

A single-sentence description is written for each MultiWOZ slot to provide slot definitions. The descriptions are based on Lin et al. [39] but with improvements in detail and grammar. Similarly, a set of 4 value examples is written by hand for each slot.

In addition to zero-shot DST, we evaluate on a 3-shot setting, for which a single set of 3 shots is collected for each MultiWOZ slot. Each shot is hand picked as a single turn and value pair from the validation split, with some light revisions made in order to shorten the number of tokens dedicated to shots.

Note that no prompt engineering process or validation experiments were performed when collecting slot descriptions, value examples, and shots. Instead, these resources were created to simply provide a clear and concise representation of each slot type, as if meant for human interpretation. This process best reflects how such resources would be initially constructed in an application setting, although prompt tuning and refinement of the descriptions and examples could be used to possibly improve performance.

6.4 Models

Experiments were conducted using T5-3B and Llama-13B-Chat base models using the sequence-to-sequence formulation shown in Figure 6.1, which is similar to the "independent" formulation of Gupta et al. [19].

T5-3B [59] is a 3 billion parameter encoder-decoder transformer model trained on a variety of sequence-to-sequence tasks such as summarization and translation.

Llama-13B-Chat is a 13 billion parameter decoder-only transformer model trained on a variety of long-form texts, then further trained on instruction data using the Reinforcement Learning from Human Feedback (RLHF) technique [50]. Due to the computational expense of its 13B parameter size, the model was quantized using QLoRA [9], which uses 4-bit `nf4` quantization, and freezes the base model parameters while only training the parameters of a Low-Rank Adapter (LoRA) [25]. A LoRA rank of 32 is used.

Experimental comparisons are made between models pretrained on synthetic data only using the data generation approach from Section 6.2, models pretrained on synthetic data and finetuned on MultiWOZ, and baseline models trained only on MultiWOZ.

Baseline Models (BL) are trained only on the MultiWOZ training split, with the test domain held out for each experiment run in accordance with the standard leave-one-out evaluation setup.

Pretrained-Only Models (PT) are trained only on synthetic data using the fully automatic data generation approach outlined in Section 6.2. We reuse the DSG5K synthetic dataset with silver DSG state update labels from Section 4.4 in line with this approach, which consists of 5,015 dialogues and 173,572 slot value pairs, each of which was extended with a description and value examples using GPT-3.5. An empty slot sampling rate of $\alpha = 0.5$ was used, resulting in training data consisting of $1/3$ empty slot examples and $2/3$ original slot value examples.

DSG-Pretrained Models (DSG) are pretrained using the presented automatic data generation approach, then finetuned in a second, separate training run on MultiWOZ similar to the baseline approach.

3-Shot Models (3S) are provided 3 shots for in-context learning during all training and testing.

6.5 Results

data	model	shots	params	avg.	attr.	hotel	rest.	taxi	train
MWOZ 2.4	IC-DST [26]	1	175B*	58.7	62.1	53.2	54.9	71.9	51.4
	RefPyDST [30]	1	175B*	68.8	74.5	56.6	68.2	68.5	76.1
	T5-PT	0	3B	18.6	29.7	10.2	36.1	5.1	12.1
	Llama-PT	0	13B†	23.6	26.7	11.4	39.7	13.9	26.9
	T5-BL	0	3B	49.2	63.2	26.0	71.7	29.8	55.8
	T5-DSG	0	3B	51.5 (+2.3)	69.1	29.9	73.2	29.2	56.2
	Llama-BL-3S	3	13B†	57.6	74.7	25.2	68.6	48.3	71.3
	Llama-BL	0	13B†	59.2	62.2	44.9	69.8	49.1	70.2
	Llama-DSG	0	13B†	65.9 (+6.7)	74.4	56.4	76.0	54.7	68.3
	Llama-DSG-3S	3	13B†	66.8 (+9.2)	76.8	48.9	78.8	53.6	76.1
MWOZ 2.1	TripPy-R [21]	0	< 1B	29.2	27.1	18.3	15.3	61.5	23.7
	LAQ [75]	0	< 1B	46.2	46.5	29.4	54.0	65.5	35.4
	D3ST [94]	0	11B	46.7	56.4	21.8	38.2	78.4	38.7
	ChatGPT [22]	0	175B	56.4	52.7	42.0	55.8	70.9	60.8
	IC-DST [26]	1	175B*	57.0	60.0	46.7	57.3	71.4	49.4
	T5-seq [19]	0	11B	64.4	76.1	28.6	69.8	87.0	60.4
	RefPyDST [30]	1	175B*	64.7	70.9	51.2	65.6	67.1	69.2
	SDT [19]	1	11B	65.9	74.4	33.9	72.0	86.4	62.9
	T5-PT	0	3B	17.2	27.3	8.9	32.8	4.9	12.6
	Llama-PT	0	13B†	23.9	28.4	11.6	37.7	13.1	27.2
	T5-BL	0	3B	44.7	57.1	22.8	60.6	30.9	52.2
	T5-DSG	0	3B	46.3 (+1.6)	61.5	27.3	62.3	29.1	51.7
	Llama-BL	0	13B†	51.8	55.4	38.8	59.0	44.8	61.2
	Llama-DSG	0	13B†	56.2 (+4.4)	63.1	43.8	64.7	48.8	60.8

Table 6.1: DST results on MultiWOZ, sorted within each division by avg. Joint Goal Accuracy (Eq. 2.1). Improvement to performance resulting from leveraging the presented data generation approach for pretraining provided in parentheses. * denotes the conjectured parameter size of OpenAI Codex. † denotes use of QLoRA.

Results indicate the effectiveness of the presented data generation approach to create diverse training resources. By pretraining on diverse data, the zero-shot Llama model saw a +6.7 point improvement on MultiWOZ 2.4. The improvement of T5 was more modest at +2.3 JGA, which is a counterintuitive result. Usually, techniques that boost performance on smaller models are more likely to disappear when a larger or stronger base model is used, rather than the reverse. A

possible explanation is that the T5 base model suffers from catastrophic forgetting due to the 2-stage training process, wherein the pretraining knowledge encoded in the base T5 model parameters is lost during the synthetic data pretraining, leading to poor finetuning performance on the MultiWOZ data. Llama-DSG is more likely to avoid this problem due to the use of LoRA, since original model parameters were frozen during training.

The results of the 3S models are concerning due to the -1.6 performance dropoff of the baseline Llama-BL-3S compared to the baseline zero-shot model. This dropoff was caused almost entirely by poor performance on the Hotel domain, due to consistent errors on the `hotel internet` slot. This may indicate an issue where the shots confused the model rather than providing supporting information. Despite this, the Llama-DSG-3S model achieved the best performance out of any model using the presented approach at 66.8 JGA. This is within 2% of the state of the art model RefPyDST [30], which uses the costly OpenAI Codex API.

The performance of the DSG-pretrained models sees a massive performance dropoff on the MultiWOZ 2.1 test set, such as the zero-shot Llama-DSG suffering -9.7 JGA. This may be unusual, as largest performance dropoff from any baseline model when comparing MultiWOZ 2.1 vs 2.4 evaluation is only -4.1 . Unfortunately, the MultiWOZ 2.4 corrections to the test set were made quite recently at the time of writing, thus most previous work only evaluates on MultiWOZ 2.1. On this benchmark, Llama-DSG performs about as well as zero-shot ChatGPT and the OpenAI Codex-based approach IC-DST. The work by [19] still holds state-of-the-art on this benchmark at an impressive 65.9 JGA, using a T5-11B model with a simple 1-shot sequence-to-sequence approach and fine-tuning all model parameters using 32 TPU machines. A question for future work is to what extent a model of this size would benefit from pretraining on diverse data with all parameters trainable.

Error	Definition	Example	#
Agent Value Miss	No value is outputted for the indicated slot, even though the information is present in the system's turns.	B: bedouin servers african food and is in the centre of town in the expensive price range . would you like their phone number , or should i book a reservation ? A: please book a reservation for wednesday at 20:00 for 5 . restaurant name: None	13
No preference	Indications of no preference are inappropriately understood, either by failing to recognize when no preference is given or by incorrectly interpreting an indication of no preference from the dialogue.	A: are there any 1 star hotel -s with free parking ? B: i am sorry , i cannot find any hotel -s that meet your criteria . would you like to search again ? A: i am looking for a guest house with free parking , preferably 1 star . B: sorry , i am not finding anything . want to change to a hotel ? A: what about guest houses ? hotel parking: any	13
Value Change	The appropriate value for the indicated slot has been updated in the dialogue turn, but the predicted value remains as the original.	A: i am looking for a place to dine in the centre that serves jamaican food. B: i am sorry , i am not finding any place that serves jamaican food in the centre of town . would you like to try another area ? A: that is fine , how about a place that serves indian food ? restaurant food: jamaican	10
Hallucination	A value is predicted for the indicated slot that does not exist in the dialogue.	A: i need a train that departs after 08:30 on friday . B: i have a train leaving cambridge arriving at birmingham new street on friday at 9:01 . would you like me to book this for you ? A: yes please , that sounds perfect . train leave at: after 16:30	9
Miss	No value is outputted for the indicated slot, even though the information is present in the user's turns.	A: are there any guest houses in the east ? B: there are 6 guest houses in the east . do you have a preference on price? A: it needs to be cheap . hotel area: None	7
Wrong Value	Information in the dialogue is incorrectly attributed to the indicated slot.	A: hey , are there any interesting attractions in towncentre today ? attraction type: interesting	6
Other	Errors not explained by any of the other error patterns.	A: i am looking for a 3 star hotel in cambridge with free wifi B: there are 5 hotel -s that meet that criteria . do you have a location or price range in mind ? A: i would like something cheap . B: unfortunately , none of the cheap hotel -s have free wifi . would you prefer to try a different price range or go with a hotel without free wifi ? A: how about a hotel with a 4 star rating ? hotel internet: no	16
Correct	The predicted value for the indicated slot is correct, but is missing from the gold annotations in MultiWOZ due to an annotation mistake.	A: i am looking for information on a hotel called warkworth house . B: i got it . may i get the day youd like to move in so i can book hotel type: hotel	26

Table 6.2: Error analysis of DSG5K Zero-shot pretrained model Llama-PT on MultiWOZ

6.6 Error Analysis

The impact of diverse DST training data is further investigated by conducting an error analysis on 100 randomly sampled errors from the Llama-PT model’s evaluation on MultiWOZ 2.4. The results of this error analysis with examples can be seen in Table 6.2. As expected, some of the errors made by this model are due to slot semantics specific to the MultiWOZ task that are difficult to encode in a single-sentence slot description. For example, the `dontcare` value (represented as `any` to the model) is a frequent source of errors, as the model consistently overpredicts it in the Hotel domain. Many errors also stem from a slot being filled with a wrong value that does indeed appear in the dialogue, but does not quite fit the specifics of the definition of the MultiWOZ slot. However, the majority of errors made are due to limitations in the training formulation using the synthetic dataset. For example, the dialogues generated by GPT-3.5 rarely include corrections or clarifications where slot value would change, resulting in consistent errors when the user speaker changes their mind or self-corrects in MultiWOZ. Also, the military time format used in MultiWOZ for time slots was a consistent source of hallucinations, as this format rarely or never appears in the synthetic DSG5K data. Finally, the model frequently missed slot values entirely, particularly when the value originated from the system travel agent speaker.

6.7 Discussion

This chapter demonstrates the critical importance of data diversity in enhancing the domain adaptability of dialogue state tracking (DST) models. By leveraging a dialogue state generation (DSG) approach, the study overcomes the limitations posed by the narrow coverage of traditional DST datasets like MultiWOZ and SGD, leading to substantial improvements in zero- and few-shot DST tasks. The significant performance boost observed underscores the efficacy of diverse pretraining data in enhancing model robustness and adaptability, with the Llama-DSG model showing a notable improvement on MultiWOZ 2.4. However, the more modest improvement seen when training T5-3B and the remaining performance gap compared to larger models with all parameters trained requires

further investigation to see the impact of diverse training data with other approaches. The presented experiment results suggest that the limited diversity of popular training data does indeed limit the ability of DST models to adapt to new domains. Although high quality data is costly to collect, use of data generation approaches such as LLM-based dialogue generation, DSG automatic state labeling, and Slot Schema Induction approaches may provide a viable way to alleviate performance bottlenecks caused by a lack of data diversity and lead to a new state-of-the-art in DST.

Chapter 7

Conclusion

The research presented in this dissertation focused on tackling the challenge of efficient domain adaptation for DST in order to facilitate the integration of conversational AI into real-world software systems, and has made several research contributions.

DST for Chat The presented chat-oriented dialogue system Emora demonstrates that DST is indeed useful not only for task-oriented dialogue systems but also for enabling rich and engaging open-domain conversations. In particular, the controllability afforded by integrating custom policy rules to modify application behavior was shown to be extremely effective. This suggests that DST may benefit from a shift in focus that targets a more diverse range of slot types and domains, or from being used explicitly as a mechanism to control dialogue system behavior.

Dialogue State for Any Domain The introduction of the Dialogue State Generation (DSG) task makes strides towards inferring dialogue state across different domains without the need for explicitly defined slots or domain adaptation resources. The presented DSG model is shown to successfully achieve domain generality when inferring slots and values for new domains, even on out-of-distribution data. This approach not only increases domain generality but also enriches the diversity of training data for DST models.

Practical Slot Induction The approach to DSG was leveraged to develop the first practical slot schema induction method DSG-I, where slots are automatically induced from a set of unlabeled dialogue data. This addresses the limitation of the DSG approach regarding the schema inconsistency of slot names. The impact of DSG-I is the reduction of laborious manual effort typically required to specify slots for DST, thus offering a more efficient solution for adapting models to new domains.

Robust DST Domain Adaptation Experiments leveraging DSG for automatically generating diverse DST training resources demonstrate that diversity of training data improves performance substantially when adapting DST models to new target domains. This demonstrates both the existing performance bottleneck caused by limited diversity in existing training resources, but also a direction for alleviating this bottleneck via diverse data generation. The presented best-performing DST model achieves competitive performance with the current SoTA LLM model on the updated MultiWOZ 2.4 benchmark, while being much more computationally efficient.

7.1 Limitations and Future Work

DSG Schema Inconsistency Although the schema consistency issue of DSG was addressed via a clustering-based SSI method, use of a clustering-based approach to slot induction requires the availability of enough unlabeled dialogue text to provide enough value candidate density to induce meaningful clusters. Furthermore, the encoding and clustering steps are not perfect, and the pipelined nature of the approach propagates DSG errors that do occur to these two steps. Thus not all DSG schema inconsistencies are resolved. Future work should look to resolve the slot schema inconsistency in a more fundamental way, perhaps through a new DSG approach that generates state inferences while jointly inducing a consistent slot schema as dialogues are processed in a streaming manner, combining the advantages of both DSG and SSI.

Noise in Synthetic DST Resources Another limitation of the presented experiments for increasing the diversity of DST training resources is the noise of the synthetic training resources. Although the

presented DSG5K data is extremely diverse, human evaluation indicates it is slightly noisy with about 15 – 20% incorrect slot-value predictions as judged by human evaluators. Therefore, although the DSG data generation approach allows creating a training resource of unprecedented diversity, it does not perfectly reflect the potential gains of increasing DST training diversity due to its noise. The automatic data generation approaches can be treated as a baseline for future work to refine and improve upon. Further refinement of diverse DST training resources, whether synthetic or collected in a real-world setting, will help estimate the impact of training diversity on DST domain adaptation more precisely.

Constraints on DST Experiments Due to constraints of time and computational resources, the DSG-based pretraining approach for DST could not be applied to the model holding the current SoTA on the MultiWOZ 2.1 benchmark. Therefore, while the diverse pretraining is likely to result in a similar boost in performance for SoTA models as the ones used in the presented experiments, this is ultimately a hypothesis to be investigated in future work.

DST Evaluation Diversity The presented exploration for improving the domain adaptation efficiency and efficacy for DST models is limited by the diversity of existing evaluation data. One of the goals of improving domain adaptation using the presented approaches is to provide improved support for niche domains and slots, but this is not evaluated due to the lack of such gold data. As demonstrated by the real-world dialogue system Emora presented in Chapter 3, fine-grained slot types that are very unlikely to be seen in other application domains can be quite important and useful for controlling the dialogue application’s behavior. However, the slot types covered in the popular MultiWOZ and SGD datasets are usually quite generic. This gap in evaluation coverage should be explored in future work, and might reveal crucial new information regarding the performance of various DST approaches.

7.2 Impact

Overall, the findings presented in this dissertation contribute to advancing the field of dialogue systems by addressing key challenges in domain adaptation and enhancing the flexibility and effectiveness of dialogue state tracking. This research illustrates the importance of DST in real-world applications, demonstrates a performance bottleneck in DST due to insufficiently diverse training resources, and explores one successful strategy for improving DST data diversity via synthetic data generation. All data and models produced during this work are made publicly available for future research and applications to make use of. These contributions facilitate the integration of conversational AI in real-world applications, taking steps towards the global improvement of software applications' efficacy and ease of use.

Bibliography

- [1] Ibrahim Aksu, Zhengyuan Liu, Min-Yen Kan, and Nancy Chen. N-Shot Learning for Augmenting Task-Oriented Dialogue State Tracking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1659–1671, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.131. URL <https://aclanthology.org/2022.findings-acl.131>.
- [2] Ibrahim Taha Aksu, Zhengyuan Liu, Min-Yen Kan, and Nancy Chen. Velocidapter: Task-oriented Dialogue Comprehension Modeling Pairing Synthetic Text Generation with Domain Adaptation. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 133–143, Singapore and Online, July 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.sigdial-1.14>.
- [3] Hedin Beattie, Lanier Watkins, William H. Robinson, Aviel Rubin, and Shari Watkins. Measuring and Mitigating Bias in AI-Chatbots. In *2022 IEEE International Conference on Assured Autonomy (ICAA)*, pages 117–123, March 2022. doi: 10.1109/ICAA52185.2022.00023.
- [4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec

- Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html>.
- [5] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1547. URL <https://aclanthology.org/D18-1547>.
- [6] Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. Zero-Shot Transfer Learning with Synthesized Data for Multi-Domain Dialogue State Tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.12. URL <https://aclanthology.org/2020.acl-main.12>.
- [7] Derek Chen, Kun Qian, and Zhou Yu. Stabilized In-Context Learning with Pre-trained Language Models for Few Shot Dialogue State Tracking. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1551–1564, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. URL <https://aclanthology.org/2023.findings-eacl.115>.
- [8] Zhi Chen, Yuncong Liu, Lu Chen, Su Zhu, Mengyue Wu, and Kai Yu. OPAL: Ontology-Aware Pretrained Language Model for End-to-End Task-Oriented Dialogue. *Transactions of the Association for Computational Linguistics*, 11: 68–84, January 2023. ISSN 2307-387X. doi: 10.1162/tacl_a_00534. URL https://direct.mit.edu/tacl/article/doi/10.1162/tacl_a_00534/114595/OPAL-Ontology-Aware-Pretrained-Language-Model-for.

- [9] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. *Advances in Neural Information Processing Systems*, 36:10088–10115, December 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1feb87871436031bdc0f2beaa62a049b-Abstract-Conference.html.
- [10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- [11] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. September 2018. URL https://openreview.net/forum?id=r1l73iRqKm&source=post_page-----.
- [12] Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. Few-NERD: A Few-shot Named Entity Recognition Dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.248. URL <https://aclanthology.org/2021.acl-long.248>.
- [13] Sarah E. Finch and Jinho D. Choi. Towards Unified Dialogue System Evaluation: A Comprehensive Analysis of Current Evaluation Protocols. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 236–

- 245, 1st virtual meeting, July 2020. Association for Computational Linguistics. URL <https://aclanthology.org/2020.sigdial-1.29>.
- [14] Sarah E. Finch, James D. Finch, Ali Ahmadvand, Ingyu, Choi, Xiangjue Dong, Ruixiang Qi, Harshita Sahijwani, Sergey Volokhin, Zihan Wang, Zihao Wang, and Jinho D. Choi. Emora: An Inquisitive Social Chatbot Who Cares For You, September 2020. URL <https://arxiv.org/abs/2009.04617v1>.
- [15] Sarah E. Finch, James D. Finch, and Jinho D. Choi. Don't Forget Your ABC's: Evaluating the State-of-the-Art in Chat-Oriented Dialogue Systems. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15044–15071, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.839. URL <https://aclanthology.org/2023.acl-long.839>.
- [16] Raefer Gabriel, Yang Liu, Anna Gottardi, Mihail Eric, Anju Khatr, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, Ali Binici, Shui Hu, Karthik Gopalakrishnan, Seokhwan Kim, Lauren Stubel, Kate Bland, Arindam Mandal, and Dilek Hakkani-Tür. Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge. In *Alexa Prize SocialBot Grand Challenge 3 Proceedings*, 2019.
- [17] Karthik Gopalakrishnan, Behnam Hedayatnia, Qinlang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. Topical-Chat: Towards Knowledge-Grounded Open-Domain Conversations. In *Interspeech 2019*, pages 1891–1895. ISCA, September 2019. doi: 10.21437/Interspeech.2019-3079. URL https://www.isca-speech.org/archive/interspeech_2019/gopalakrishnan19_interspeech.html.
- [18] Prakhar Gupta, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. DialFact: A Benchmark for Fact-Checking in Dialogue. In *Proceedings of the 60th Annual Meeting of the Association*

- for Computational Linguistics (Volume 1: Long Papers)*, pages 3785–3801, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.263. URL <https://aclanthology.org/2022.acl-long.263>.
- [19] Raghav Gupta, Harrison Lee, Jeffrey Zhao, Yuan Cao, Abhinav Rastogi, and Yonghui Wu. Show, Don't Tell: Demonstrations Outperform Descriptions for Schema-Guided Task-Oriented Dialogue. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4541–4549, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.336. URL <https://aclanthology.org/2022.naacl-main.336>.
- [20] Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauer, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking. In Olivier Pietquin, Smaranda Muresan, Vivian Chen, Casey Kennington, David Vandyke, Nina Dethlefs, Koji Inoue, Erik Ekstedt, and Stefan Ultes, editors, *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sigdial-1.4. URL <https://aclanthology.org/2020.sigdial-1.4>.
- [21] Michael Heck, Nurul Lubis, Carel van Niekerk, Shutong Feng, Christian Geishauer, Hsien-Chin Lin, and Milica Gašić. Robust Dialogue State Tracking with Weak Supervision and Sparse Data. *Transactions of the Association for Computational Linguistics*, 10:1175–1192, 2022. doi: 10.1162/tacl_a_00513. URL <https://aclanthology.org/2022.tacl-1.68>. Place: Cambridge, MA Publisher: MIT Press.
- [22] Michael Heck, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauer, Hsien-Chin Lin, Carel van Niekerk, and Milica Gašić. ChatGPT for Zero-

- shot Dialogue State Tracking: A Solution or an Opportunity?, June 2023. URL <http://arxiv.org/abs/2306.01386>. arXiv:2306.01386 [cs].
- [23] Ryuichiro Higashinaka, Masahiro Araki, Hiroshi Tsukahara, and Masahiro Mizukami. Integrated taxonomy of errors in chat-oriented dialogue systems. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 89–98, Singapore and Online, July 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.sigdial-1.10>.
- [24] Or Honovich, Leshem Choshen, Roei Aharoni, Ella Neeman, Idan Szpektor, and Omri Abend. Q2: Evaluating Factual Consistency in Knowledge-Grounded Dialogues via Question Generation and Question Answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7856–7870, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.619. URL <https://aclanthology.org/2021.emnlp-main.619>.
- [25] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. March 2022. URL <https://openreview.net/forum?id=nZeVKeeFYf9>.
- [26] Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. In-Context Learning for Few-Shot Dialogue State Tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.findings-emnlp.193>.
- [27] Vojtěch Hudeček, Ondřej Dušek, and Zhou Yu. Discovering Dialogue Slots with Weak Supervision. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2430–2442, Online, August 2021. Associa-

- tion for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.189. URL <https://aclanthology.org/2021.acl-long.189>.
- [28] Megha Jhunjhunwala, Caleb Bryant, and Pararth Shah. Multi-Action Dialog Policy Learning with Interactive Human Teaching. In Olivier Pietquin, Smaranda Muresan, Vivian Chen, Casey Kennington, David Vandyke, Nina Dethlefs, Koji Inoue, Erik Ekstedt, and Stefan Ultes, editors, *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 290–296, 1st virtual meeting, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sigdial-1.36. URL <https://aclanthology.org/2020.sigdial-1.36>.
- [29] Sungdong Kim, Minsuk Chang, and Sang-Woo Lee. NeuralWOZ: Learning to Collect Task-Oriented Dialogue via Model-Based Simulation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3704–3717, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.287. URL <https://aclanthology.org/2021.acl-long.287>.
- [30] Brendan King and Jeffrey Flanigan. Diverse Retrieval-Augmented In-Context Learning for Dialogue State Tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5570–5585, Toronto, Canada, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.344. URL <https://aclanthology.org/2023.findings-acl.344>.
- [31] Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html.

- [32] Wai-Chung Kwan, Hong-Ru Wang, Hui-Min Wang, and Kam-Fai Wong. A Survey on Recent Advances and Challenges in Reinforcement Learning Methods for Task-oriented Dialogue Policy Learning. *Machine Intelligence Research*, 20(3):318–334, June 2023. ISSN 2731-5398. doi: 10.1007/s11633-022-1347-y. URL <https://doi.org/10.1007/s11633-022-1347-y>.
- [33] Chun-Mao Lai, Ming-Hao Hsu, Chao-Wei Huang, and Yun-Nung Chen. Controllable User Dialogue Act Augmentation for Dialogue State Tracking. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 53–61, Edinburgh, UK, September 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.sigdial-1.5>.
- [34] S. Larsson and D. R. Traum. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *NLE*, 6(3 & 4):323–340, Sep 2000. ISSN 13513249. doi: 10.1017/S1351324900002539.
- [35] Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. SGD-X: A Benchmark for Robust Generalization in Schema-Guided Dialogue Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10938–10946, June 2022. ISSN 2374-3468. doi: 10.1609/aaai.v36i10.21341. URL <https://ojs.aaai.org/index.php/AAAI/article/view/21341>. Number: 10.
- [36] Margaret Li, Jason Weston, and Stephen Roller. ACUTE-EVAL: Improved Dialogue Evaluation with Optimized Questions and Multi-turn Comparisons, September 2019. URL <https://arxiv.org/abs/1909.03087v1>.
- [37] Zekang Li, Jinchao Zhang, Zhengcong Fei, Yang Feng, and Jie Zhou. Addressing Inquiries about History: An Efficient and Practical Framework for Evaluating Open-domain Chatbot Consistency. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1057–1067, Online, August 2021. Association for Computational Linguistics.

- doi: 10.18653/v1/2021.findings-acl.91. URL <https://aclanthology.org/2021.findings-acl.91>.
- [38] Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics - ACL '04*, pages 605–es, Barcelona, Spain, 2004. Association for Computational Linguistics. doi: 10.3115/1218955.1219032. URL <http://portal.acm.org/citation.cfm?doid=1218955.1219032>.
- [39] Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue StateTracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5640–5648, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.448. URL <https://aclanthology.org/2021.naacl-main.448>.
- [40] Navonil Majumder, Pengfei Hong, Shanshan Peng, Jiankun Lu, Deepanway Ghosal, Alexander Gelbukh, Rada Mihalcea, and Soujanya Poria. MIME: MIMicking Emotions for Empathetic Response Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8968–8979, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.721. URL <https://aclanthology.org/2020.emnlp-main.721>.
- [41] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, March 2017. ISSN 2475-9066. doi: 10.21105/joss.00205. URL <https://joss.theoj.org/papers/10.21105/joss.00205>.
- [42] Shikib Mehri and Maxine Eskenazi. Unsupervised Evaluation of Interactive Dialog with DialoGPT. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on*

- Discourse and Dialogue*, pages 225–235, 1st virtual meeting, July 2020. Association for Computational Linguistics. URL <https://aclanthology.org/2020.sigdial-1.28>.
- [43] Shikib Mehri and Maxine Eskenazi. USR: An Unsupervised and Reference Free Evaluation Metric for Dialog Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 681–707, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.64. URL <https://aclanthology.org/2020.acl-main.64>.
- [44] Shikib Mehri, Yasemin Altun, and Maxine Eskenazi. LAD: Language Models as Data for Zero-Shot Dialog. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 595–604, Edinburgh, UK, September 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.sigdial-1.55>.
- [45] Chuan Meng, Pengjie Ren, Zhumin Chen, Weiwei Sun, Zhaochun Ren, Zhaopeng Tu, and Maarten de Rijke. DukeNet: A Dual Knowledge Interaction Network for Knowledge-Grounded Conversation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20*, pages 1151–1160, New York, NY, USA, July 2020. Association for Computing Machinery. ISBN 978-1-4503-8016-4. doi: 10.1145/3397271.3401097. URL <https://doi.org/10.1145/3397271.3401097>.
- [46] Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. Generating Training Data with Language Models: Towards Zero-Shot Language Understanding. *Advances in Neural Information Processing Systems*, 35:462–477, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0346c148ba1c21c6b4780a961ea141dc-Abstract-Conference.html.
- [47] Qingkai Min, Libo Qin, Zhiyang Teng, Xiao Liu, and Yue Zhang. Dialogue state induction

- using neural latent variable models. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI'20*, pages 3845–3852, Yokohama, Yokohama, Japan, January 2021. ISBN 978-0-9992411-6-5.
- [48] Biswesh Mohapatra, Gaurav Pandey, Danish Contractor, and Sachindra Joshi. Simulated Chats for Building Dialog Systems: Learning to Generate Conversations from Instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1190–1203, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.103. URL <https://aclanthology.org/2021.findings-emnlp.103>.
- [49] Yixin Nie, Mary Williamson, Mohit Bansal, Douwe Kiela, and Jason Weston. I like fish, especially dolphins: Addressing Contradictions in Dialogue Modeling. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1699–1713, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.134. URL <https://aclanthology.org/2021.acl-long.134>.
- [50] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, December 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.
- [51] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on*

- Association for Computational Linguistics - ACL '02*, page 311, Philadelphia, Pennsylvania, 2001. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL <http://portal.acm.org/citation.cfm?doid=1073083.1073135>.
- [52] Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. Soloist: Building Task Bots at Scale with Transfer Learning and Machine Teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824, 2021. doi: 10.1162/tacl_a_00399. URL <https://aclanthology.org/2021.tacl-1.49>. Place: Cambridge, MA Publisher: MIT Press.
- [53] Vitou Phy, Yang Zhao, and Akiko Aizawa. Deconstruct to Reconstruct a Configurable Evaluation Metric for Open-Domain Dialogue Systems. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4164–4178, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.368. URL <https://aclanthology.org/2020.coling-main.368>.
- [54] Matt Post. A Call for Clarity in Reporting BLEU Scores. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6319. URL <https://aclanthology.org/W18-6319>.
- [55] Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. AGIF: An Adaptive Graph-Interactive Framework for Joint Multiple Intent Detection and Slot Filling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1807–1816, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.163. URL <https://aclanthology.org/2020.findings-emnlp.163>.

- [56] Liang Qiu, Chien-Sheng Wu, Wenhao Liu, and Caiming Xiong. Structure Extraction in Task-Oriented Dialogues with Slot Clustering, March 2022. URL <http://arxiv.org/abs/2203.00073>. arXiv:2203.00073 [cs].
- [57] Jun Quan and Deyi Xiong. Effective Data Augmentation Approaches to End-to-End Task-Oriented Dialogue. In *2019 International Conference on Asian Language Processing (IALP)*, pages 47–52, November 2019. doi: 10.1109/IALP48816.2019.9037690. URL <https://ieeexplore.ieee.org/abstract/document/9037690>.
- [58] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.
- [59] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):140:5485–140:5551, January 2020. ISSN 1532-4435.
- [60] Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. Towards Empathetic Open-domain Conversation Models: A New Benchmark and Dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5370–5381, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1534. URL <https://aclanthology.org/P19-1534>.
- [61] Hannah Rashkin, David Reitter, Gaurav Singh Tomar, and Dipanjan Das. Increasing Faithfulness in Knowledge-Grounded Dialogue with Controllable Features. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 704–718,

- Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.58. URL <https://aclanthology.org/2021.acl-long.58>.
- [62] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8689–8696, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i05.6394. URL <https://ojs.aaai.org/index.php/AAAI/article/view/6394>. Number: 05.
- [63] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL <https://aclanthology.org/D19-1410>.
- [64] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. Recipes for Building an Open-Domain Chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.24. URL <https://aclanthology.org/2021.eacl-main.24>.
- [65] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. Recipes for Building an Open-Domain Chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.24. URL <https://aclanthology.org/2021.eacl-main.24>.

- [66] Sahand Sabour, Chujie Zheng, and Minlie Huang. CEM: Commonsense-Aware Empathetic Response Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10): 11229–11237, June 2022. ISSN 2374-3468. doi: 10.1609/aaai.v36i10.21373. URL <https://ojs.aaai.org/index.php/AAAI/article/view/21373>. Number: 10.
- [67] Manuela Sanguinetti, Alessandro Mazzei, Viviana Patti, Marco Scalerandi, Dario Mana, and Rossana Simeoni. Annotating Errors and Emotions in Human-Chatbot Interactions in Italian. In *Proceedings of the 14th Linguistic Annotation Workshop*, pages 148–159, Barcelona, Spain, December 2020. Association for Computational Linguistics. URL <https://aclanthology.org/2020.law-1.14>.
- [68] Sashank Santhanam, Behnam Hedayatnia, Spandana Gella, Aishwarya Padmakumar, Seokhwan Kim, Yang Liu, and Dilek Hakkani-Tur. Rome was built in 1776: A Case Study on Factual Correctness in Knowledge-Grounded Response Generation, October 2021. URL <http://arxiv.org/abs/2110.05456>. arXiv:2110.05456 [cs].
- [69] Jamin Shin, Hangeol Yu, Hyeongdon Moon, Andrea Madotto, and Juneyoung Park. Dialogue Summaries as Dialogue States (DS2), Template-Guided Summarization for Few-shot Dialogue State Tracking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3824–3846, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.302. URL <https://aclanthology.org/2022.findings-acl.302>.
- [70] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval Augmentation Reduces Hallucination in Conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.320. URL <https://aclanthology.org/2021.findings-emnlp.320>.
- [71] Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, and Y-Lan Boureau.

- Can You Put it All Together: Evaluating Conversational Agents' Ability to Blend Skills. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2021–2030, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.183. URL <https://aclanthology.org/2020.acl-main.183>.
- [72] Adam Summerville, Jordan Hashemi, James Ryan, and William Ferguson. How to Tame Your Data: Data Augmentation for Dialog State Tracking. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 32–37, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.nlp4convai-1.4. URL <https://www.aclweb.org/anthology/2020.nlp4convai-1.4>.
- [73] Hao Sun, Guangxuan Xu, Jiawen Deng, Jiale Cheng, Chujie Zheng, Hao Zhou, Nanyun Peng, Xiaoyan Zhu, and Minlie Huang. On the Safety of Conversational Models: Taxonomy, Dataset, and Benchmark. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3906–3923, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.308. URL <https://aclanthology.org/2022.findings-acl.308>.
- [74] Jianheng Tang, Tiancheng Zhao, Chenyan Xiong, Xiaodan Liang, Eric Xing, and Zhiting Hu. Target-Guided Open-Domain Conversation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5624–5634, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1565. URL <https://aclanthology.org/P19-1565>.
- [75] Diogo Tavares, David Semedo, Alexander Rudnicky, and Joao Magalhaes. Learning to Ask Questions for Zero-shot Dialogue State Tracking. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23*, pages 2118–2122, New York, NY, USA, July 2023. Association for Computing Machinery.

ISBN 978-1-4503-9408-6. doi: 10.1145/3539618.3592010. URL <https://dl.acm.org/doi/10.1145/3539618.3592010>.

- [76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [77] Dazhen Wan, Zheng Zhang, Qi Zhu, Lizi Liao, and Minlie Huang. A Unified Dialogue User Simulator for Few-shot Data Augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3788–3799, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.277. URL <https://aclanthology.org/2022.findings-emnlp.277>.
- [78] Qingyue Wang, Liang Ding, Yanan Cao, Yibing Zhan, Zheng Lin, Shi Wang, Dacheng Tao, and Li Guo. Divide, Conquer, and Combine: Mixture of Semantic-Independent Experts for Zero-Shot Dialogue State Tracking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2048–2061, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.114. URL <https://aclanthology.org/2023.acl-long.114>.
- [79] Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 17, 2011.
- [80] Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. Dialogue Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1363. URL <https://aclanthology.org/P19-1363>.
- [81] Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. Symbolic Knowledge Distillation: from Gen-

- eral Language Models to Commonsense Models. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.341. URL <https://aclanthology.org/2022.naacl-main.341>.
- [82] Jason Weston and Kurt Shuster. Blender Bot 2.0: An open source chatbot that builds long-term memory and searches the internet, July 2021.
- [83] Seungpil Won, Heeyoung Kwak, Joongbo Shin, Janghoon Han, and Kyomin Jung. BREAK: Breaking the Dialogue State Tracking Barrier with Beam Search and Re-ranking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2832–2846, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.159. URL <https://aclanthology.org/2023.acl-long.159>.
- [84] Yangjun Wu, Han Wang, Dongxiang Zhang, Gang Chen, and Hao Zhang. Incorporating Instructional Prompts into a Unified Generative Framework for Joint Multiple Intent Detection and Slot Filling. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 7203–7208, Gyeongju, Republic of Korea, October 2022. International Committee on Computational Linguistics. URL <https://aclanthology.org/2022.coling-1.631>.
- [85] Yuxia Wu, Lizi Liao, Xueming Qian, and Tat-Seng Chua. Semi-supervised New Slot Discovery with Incremental Clustering. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6207–6218, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.462. URL <https://aclanthology.org/2022.findings-emnlp.462>.

- [86] Hongyan Xie, Haoxiang Su, Shuangyong Song, Hao Huang, Bo Zou, Kun Deng, Jianghua Lin, Zhihui Zhang, and Xiaodong He. Correctable-DST: Mitigating Historical Context Mismatch between Training and Inference for Improved Dialogue State Tracking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 876–889, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.emnlp-main.56>.
- [87] Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. Bot-Adversarial Dialogue for Safe Conversational Agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2950–2968, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.235. URL <https://aclanthology.org/2021.naacl-main.235>.
- [88] Jing Xu, Arthur Szlam, and Jason Weston. Beyond Goldfish Memory: Long-Term Open-Domain Conversation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5180–5197, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.356. URL <https://aclanthology.org/2022.acl-long.356>.
- [89] Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. MultiWOZ 2.4: A Multi-Domain Task-Oriented Dialogue Dataset with Essential Annotation Corrections to Improve State Tracking Evaluation. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK, September 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.sigdial-1.34>.
- [90] Fanghua Ye, Xi Wang, Jie Huang, Shenghui Li, Samuel Stern, and Emine Yilmaz. MetaAS-SIST: Robust Dialogue State Tracking with Meta Learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1157–1169, Abu

- Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.emnlp-main.76>.
- [91] Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. Dialog State Tracking with Reinforced Data Augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9474–9481, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i05.6491. URL <https://ojs.aaai.org/index.php/AAAI/article/view/6491>. Number: 05.
- [92] Dian Yu, Mingqiu Wang, Yuan Cao, Izhak Shafran, Laurent Shafey, and Hagen Soltau. Un-supervised Slot Schema Induction for Task-oriented Dialog. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1174–1193, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.86. URL <https://aclanthology.org/2022.naacl-main.86>.
- [93] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SkeHuCVFDr>.
- [94] Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. Description-Driven Task-Oriented Dialog Modeling, January 2022. URL <http://arxiv.org/abs/2201.08904>. arXiv:2201.08904 [cs].
- [95] Peixiang Zhong, Di Wang, Pengfei Li, Chen Zhang, Hao Wang, and Chunyan Miao. CARE: Commonsense-Aware Emotional Response Generation with Latent Concepts. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14577–14585, May 2021. ISSN 2374-3468. doi: 10.1609/aaai.v35i16.17713. URL <https://ojs.aaai.org/index.php/AAAI/article/view/17713>. Number: 16.
- [96] Pei Zhou, Karthik Gopalakrishnan, Behnam Hedayatnia, Seokhwan Kim, Jay Pujara, Xiang

- Ren, Yang Liu, and Dilek Hakkani-Tur. Commonsense-Focused Dialogues for Response Generation: An Empirical Study. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 121–132, Singapore and Online, July 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.sigdial-1.13>.
- [97] Pei Zhou, Karthik Gopalakrishnan, Behnam Hedayatnia, Seokhwan Kim, Jay Pujara, Xiang Ren, Yang Liu, and Dilek Hakkani-Tur. Think Before You Speak: Explicitly Generating Implicit Commonsense Knowledge for Response Generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1237–1252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.88. URL <https://aclanthology.org/2022.acl-long.88>.