KNOWLEDGE ENHANCED TASK-ORIENTED DIALOGUE SYSTEMS

YUE FENG

UNIVERSITY COLLEGE LONDON

DEPARTMENT OF COMPUTER SCIENCE

Submitted to University College London (UCL) in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy.

Primary supervisor: **EMINE YILMAZ**

Secondary supervisor: **ALDO LIPANI**

Internal Examiner: **SHI ZHOU**

External Examiner: **HAMED ZAMANI**

Thesis submission date: **15th May, 2024**

Declaration

I, **Yue Feng**, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Yue Feng
London, United Kingdom
15th May, 2024

Abstract

Task-oriented dialogue systems have gained significant attention in both academia and industry for their practical utility in assisting users with specific tasks within defined domains. Unlike open-domain systems, which focus on maximizing user engagement, task-oriented systems prioritize task completion. Existing research in this field can be broadly categorized into pipeline methods and end-to-end approaches. While pipeline methods compartmentalize the system into modules such as Natural Language Understanding (NLU), Natural Language Generation (NLG), and User satisfaction Modeling (USM). End-to-end approaches utilize a single model for seamless processing. However, integrating domain knowledge, including domain schema and domain documents, poses a significant challenge, hindering the effectiveness of these systems. With the emergence of large language models (LLMs), various strategies have been proposed to integrate common sense knowledge from LLMs to enhance system performance. This thesis explores different methodologies for integrating domain knowledge and common sense knowledge into task-oriented dialogue systems. Specifically, (1) we explore how to use the task schema knowledge to track dialogue state to enhance the natural language understanding ability of the task-oriented dialogue system; (2) we explore how to use task document knowledge to generate system responses to clarify users' needs and user profile; (3) we explore how to effectively estimate the user satisfaction using domain knowledge to better simulate the user and evaluate the performance of the task-oriented dialogue system; (4) we explore how to use common scene knowledge in the LLMs to empower the end-to-end task-oriented dialogue system. Each methodology is discussed in detail, highlighting its contributions and experimental results.

Impact Statement

This thesis contributes to the advancement of task-oriented dialogue systems by addressing critical challenges in integrating knowledge and enhancing system performance. Through the exploration of various methodologies, including schema-guided natural language understanding (NLU), document-guided natural language generation (NLG), schema-guided user satisfaction modeling, and large language model (LLM)-empowered end-to-end methods, this research significantly impacts both academia and industry in the following ways:

1. Enhanced System Performance: By leveraging different types of knowledge, such as domain knowledge and common sense knowledge, the proposed methodologies improve the effectiveness and efficiency of task-oriented dialogue systems. This leads to more accurate understanding of user intents, more informative and personalized responses, and higher levels of user satisfaction.

2. Practical Applications: The developed methodologies have practical applications in various domains, including customer service, healthcare, e-commerce, and education. Task-oriented dialogue systems equipped with these advancements can assist users in completing tasks more efficiently and effectively, leading to improved user experiences and increased productivity.

3. Research Advancements: This thesis contributes to the body of knowledge in the field of natural language processing (NLP) and human-computer interaction (HCI) by introducing novel approaches for integrating knowledge into dialogue systems. The insights gained from this research can inspire further studies in related areas and drive innovation in dialogue system design and development.

4. Industry Adoption: The methodologies proposed in this thesis have the potential to be adopted by industry practitioners to enhance the capabilities of existing task-oriented dialogue systems. By incorporating these advancements into real-world applications, companies can deliver more intelligent and user-friendly conversational interfaces to their customers, leading to improved customer satisfaction and loyalty.

5. Ethical Considerations: This research emphasizes the importance of ethical and responsible AI development in the design and deployment of task-oriented dialogue systems. By addressing ethical considerations and societal impacts, this work promotes the development of AI technologies that prioritize user privacy, fairness, and transparency.

In summary, this thesis has a significant impact on the advancement of task-oriented dialogue systems, offering novel methodologies that improve system performance, facilitate practical applications, contribute to research advancements, drive industry adoption, and promote ethical AI development.

Acknowledgements

I would like to express my sincere gratitude to everyone who has contributed to the completion of this thesis.

First and foremost, I am deeply grateful to my advisor, Prof. Emine Yilmaz and Prof. Aldo Lipani, for their guidance, support, and mentorship throughout this research endeavor. Their expertise, encouragement, and invaluable feedback have been instrumental in shaping this thesis and my academic growth.

I would also like to acknowledge my group members and collaborators for their collaboration, insights, and camaraderie. Their contributions have enriched my research experience and made this thesis possible.

I extend my heartfelt thanks to my family for their unwavering love, encouragement, and understanding. Their constant support and belief in my abilities have been my motivation and source of strength throughout this journey.

Last but not least, I would like to thank all my friends for their encouragement, inspiration, and moral support.

This thesis would not have been possible without the contributions and support of each and every individual mentioned above. Thank you for being part of this journey.

Contents

Table of Acronyms **9**

List of Figures **10**

List of Tables **13**

1 Introduction **16**
1.1 Task-oriented Dialogue Systems 16
1.2 Research Questions 16
1.3 Significance 17
1.4 Contributions 18
1.5 Publication List 20
1.6 A Readers Guide 21

2 Related Work **22**
2.1 Task-oriented Dialogue Systems 22
2.2 Natural Language Understanding for Task-oriented Dialogue Systems 24
2.3 Natural Language Generation for Task-Oriented Dialogue Systems 26
2.4 User Satisfaction Modeling for Task-Oriented Dialogue Systems 27
2.5 End-to-End Methods for Task-Oriented Dialogue Systems . . 29
2.6 Clarifying Question Generation and Selection 31

3 Schema-Guided Natural Language Understanding for Task-Oriented Dialogues **32**
3.1 Introduction 32
3.2 Seq-to-Seq Dialogue State Tracking 34
3.2.1 Utterance Encoder 34
3.2.2 Schema Encoder 35
3.2.3 Utterance-Schema Attender 36
3.2.4 State Decoder 36
3.2.5 Training 37
3.3 Experiments 37
3.3.1 Datasets 37
3.3.2 Baselines and Variants 38
3.3.3 Evaluation Measures 38
3.3.4 Training 39
3.4 Results and Discussion 39
3.4.1 Experimental Results 39
3.4.2 Ablation Study 40

3.4.3 Discussions . . . 43
3.5 Summary . . . 45

4 Relation-Guided Natural Language Understanding for Task-Oriented Dialogues 46
4.1 Introduction . . . 46
4.2 Relations among Domains and Slots . . . 48
4.3 Dynamic Schema Graph Fusion Network . . . 49
4.3.1 Dialogue Utterance Encoder . . . 50
4.3.2 Schema Graph Encoder . . . 50
4.3.3 Schema Graph Evolving Network . . . 51
4.3.4 Dialogue State Decoder . . . 52
4.3.5 Training and Inference . . . 53
4.4 Experiments . . . 53
4.4.1 Datasets . . . 53
4.4.2 Baselines . . . 53
4.4.3 Evaluation Measures . . . 54
4.4.4 Training . . . 54
4.5 Results and Discussion . . . 54
4.5.1 Ablation Study . . . 55
4.5.2 Further Analysis . . . 57
4.6 Summary . . . 60

5 Document-Guided Natural Language Generation for Task-Oriented Dialogues 63
5.1 Introduction . . . 63
5.2 Problem Formalization . . . 65
5.2.1 The Ask Paradigm in Task-Oriented Dialogue Systems 65
5.2.2 Notations and Problem Statement . . . 66
5.3 Multi-Attention Large Language Model . . . 67
5.3.1 Text Encoder . . . 68
5.3.2 Answer Confidence Embeddings Network . . . 68
5.3.3 Natural Language Decoder . . . 69
5.3.4 Training . . . 69
5.4 Data Collection . . . 69
5.4.1 User Profile Construction . . . 70
5.4.2 Dataset Quality Check . . . 71
5.5 Experiments . . . 71
5.5.1 Experiments Setup . . . 71
5.5.2 Experimental Results . . . 72
5.6 Discussions . . . 73
5.6.1 Ablation Study . . . 73
5.6.2 Case Study . . . 74
5.6.3 Utility of Clarification Questions . . . 75
5.6.4 Necessary of Clarification Questions . . . 75
5.6.5 Impact of User Request Length . . . 76
5.6.6 Impact of User Profile . . . 76
5.7 Summary . . . 77

6 Schema-Guided User Satisfaction Modeling for Task-Oriented Dialogues 78
6.1 Introduction . . . 78
6.2 Schema-guided User Satisfaction Modeling . . . 80
6.2.1 Text Encoder . . . 81
6.2.2 Task Attribute Fulfillment Representation Layer . . . 81
6.2.3 Task Attribute Importance Predictor . . . 81
6.2.4 User Satisfaction Predictor . . . 82
6.2.5 Training . . . 83
6.3 Experimental Setup . . . 83
6.3.1 Datasets . . . 83
6.3.2 Baselines and SG-USM Variants . . . 84
6.3.3 Evaluation Metrics . . . 85
6.3.4 Training . . . 85
6.4 Experimental Results . . . 85
6.4.1 Overall Performance . . . 85
6.4.2 Ablation Study . . . 86
6.4.3 Discussion . . . 88
6.5 Summary . . . 89

7 Large Language Model Empowered End-to-End Task-Oriented Dialogue Systems 90
7.1 Introduction . . . 90
7.2 Framework . . . 93
7.2.1 Sub-Task Detection . . . 93
7.2.2 Model Matching . . . 96
7.2.3 Sub-Task Execution . . . 96
7.2.4 Response Generation . . . 96
7.3 Reinforcement Learning from CRSs Performance Feedback . 97
7.4 Experiments . . . 98
7.4.1 Dataset . . . 98
7.4.2 Baselines . . . 99
7.4.3 Evaluation Measures . . . 99
7.4.4 Implementation Details . . . 100
7.5 Experimental Results and Discussion . . . 101
7.5.1 Overall Performance . . . 101
7.5.2 Ablation Study . . . 101
7.5.3 Mechanisms to Instruct LLM . . . 105
7.5.4 Case Study . . . 105
7.6 Summary . . . 106

8 Future Work 107

9 Conclusion 109

Table of Acronyms

Notation	Description
TOD	Task-Oriented Dialogue Systems.
NLU	Natural Language Understanding.
NLG	Natural Language Generation.
USM	User Satisfaction Modeling.
LLMs	Large Language Models.

List of Figures

2.1 General framework of a pipeline task-oriented dialog system. . 22
2.2 Framework of end-to-end dialog systems: Initially, the system encodes the natural language context to derive latent variables, which subsequently aid in the knowledge base (KB) querying. Subsequently, leveraging these latent variables and query results, the decoder generates a natural language response. . . 29

3.1 An example of dialogue state tracking. Given a dialogue history that contains user utterances and system utterances, and descriptions of schema that contain all possible intents and slot-value pairs, a dialogue state for the current turn is created which is represented by intents and slot-value pairs. There are slot values obtained from the schema (categorical) as well as slot values extracted from the utterances (non-categorical). #4, #6, etc denote pointers. 33
3.2 The architecture of Seq2Seq-DU, containing utterance encoder, schema encoder, utterance-schema attender, and state decoder. 35
3.3 Training losses of Seq2Seq-DU on all training datasets. 39
3.4 Accuracies of Seq2Seq-DU in terms of Join GA / Slot F1 on all test sets. 40
3.5 Ablation study results of Seq2Seq-DU with respect to BERT, attention, and pointer generation on SGD and MultiWOZ2.2. 42

4.1 An example of DST. Given the schemata for all domains, the slot values are extracted from the user and system utterances (e.g., spans highlighted with the same color in the figure). The dialogue state of each turn is represented as a set of slot-value pairs. Among the domains and slots, there are prior slot-domain membership relations which are expressed in the predefined schemata, and also dialogue-aware dynamic slot relations which depend on the dialogue context (e.g., co-reference, co-update, and co-occurrence). 47
4.2 The architecture of DSGFNet, which contains a dialogue utterance encoder, a schema graph encoder, a schema graph evolving network, and a dialogue state decoder. 50
4.3 F1 and Accuracy of DSGFNet and BERT for dynamic relation prediction on unseen domains SGD, all domains of SGD, MultiWOZ2.2 and MultiWOZ2.1. 57
4.4 Performance comparison *w.r.t.* the layer of propagation on the schema graph. 61

4.5 Performance comparison *w.r.t.* the number of dialogue turns used in the schema-dialogue fusion layer. 61

4.6 Performance comparison *w.r.t.* the layer of MLP in the dynamic slot relation completion layer. 62

4.7 Performance comparison *w.r.t.* the balance coefficient in the loss function. 62

5.1 An example of task-oriented information-seeking dialogue. A user needs to search for information related to a specific task. Due to the unclear user request and user profile, the task-oriented dialogue system should ask clarification questions to clarify the user request and the user profile based on the task knowledge to provide the answer. 64

5.2 The Ask Paradigm in Task-Oriented Dialogue Systems. 66

5.3 The architecture of MAS2S for information seeking on task-oriented dialogues. 67

5.4 The process of user profile construction. We follow a three-step strategy as follows: (1) Sample dialogue turns; (2) Generate user profile; and (3) Correct grammatical errors through crowdsourcing. 69

5.5 Ablation study of MAS2S with respect to LLMs, and confidence embeddings network on ClariT. 72

5.6 Case study based on MAS2S and the best baseline UBAR on ClariT. The generated system response in green is correctly predicted, while the generated system response in red is incorrectly predicted. 74

5.7 Impact of number of clarification questions on the performance of Success of MAS2S and UBAR. 75

5.8 Impact of user request length on the performance of Success of MAS2S. 76

6.1 Task-oriented dialogue system has a predefined schema for each task, which is composed of a set of task attributes. In a dialogue, the user's task goal is encoded by the task attribute and value pairs. The user is satisfied with the service when the provided solution fulfills the user's preferences for the task attributes. 79

6.2 Performance of SG-USM by ablating the task attribute importance and task attribute fulfillment components across datasets. 86

6.3 Case study on SG-USM and USDA on SGD dataset. The yellow star represents the importance of task attributes. The texts in green are the users' preferences for the task attributes. The texts in red are the attributes of the provided solutions. . . 87

6.4 Performance of SG-USM trained with different numbers of unlabeled dialogues on MWOZ, SGD, ReDial, and JDDC datasets. 88

7.1 An example of a conversational recommender system (CRS). A conversational recommendation typically consists of multiple sub-tasks. A CRS must have the ability for sub-task management and sub-task resolution. It also needs to generate responses to interact with users. 91
7.2 The framework of CRSAgent. It consists of an LLM as the controller and numerous expert models as collaborative executors. The LLM rst detects the sub-task. Then it selects a suitable expert model for the sub-task. After that, the sub-task is assigned to the expert model to get the results. Finally, LLM collects the results from the expert model and generates responses to the user. The Reinforcement Learning from CRSs Performance Feedback (RLPF) is used to adapt the LLM to CRSs. 93
7.3 Ablation study of CRSAgent with respect to the sub-task management, expert models, and RLPF on GoRecDial dataset. . . 103
7.4 Ablation study of CRSAgent with respect to the sub-task management, expert models, and RLPF on TG-ReDial dataset. . . 104
7.5 Case study of CRSAgent 106

List of Tables

2.1 An example of dialog act for an utterance in the restaurant reservation domain. 24

3.1 Descriptions for a dialogue schema. Two combined descriptions are used for describing an intent, a slot, or a value in the schema. 35
3.2 Statistics of datasets in experiments. Numbers are those of training datasets. 38
3.3 Accuracies of Seq2Seq-DU and baselines on SGD, MultiWOZ2.2 and MultiWOZ2.1 datasets. Seq2Seq-DU outperforms baselines in terms of all metrics. 41
3.4 Accuracies of Seq2Seq-DU and baselines on WOZ2.0 and DSTC2 datasets. Seq2Seq-DU-w/oSchema performs significantly better than the baselines. 41
3.5 Accuracies of Seq2Seq-DU and baselines on M2M dataset. Seq2Seq-DU-w/oSchema significantly outperforms the baselines. 41
3.6 Accuracies of Seq2Seq-DU and baselines on ATIS and SNIPS datasets. Seq2Seq-DU-SeqLabel performs comparably with Joint BERT. 41
3.7 Case study on Seq2Seq-DU and SGD-baseline on SGD and MultiWOZ2.2. The first example is from SGD and the second is from MultiWOZ2.2. The underlined slot-value pairs represent the ground truth states. The slot-value pairs in blue are correctly predicted ones, while the slot-value pairs in red are incorrectly predicted ones. 43
3.8 Accuracy of Seq2Seq-DU in each domain on SGD test set. Domains marked with * are those for which the schemas in the test set are not present in the training set. Domains marked with ** have both the unseen and seen schemas. For other domains, the schemas in the test set are also seen in the training set. 44
3.9 Accuracies of Seq2Seq-DU and baselines with respect to categorical and non-categorical slots on SGD and MultiWOZ2.2. . 44

4.1 Characteristics of the datasets in experiments. The numbers provided are for the training sets of the corresponding datasets. 53
4.2 Joint GA of DSGFNet and baselines in unseen domains and all domains on SGD dataset. DSGFNet significantly improves over the best baseline Seq2Seq-DU (two-sided paired t-test, $p < 0.05$). 55

4.3 Joint GA of DSGFNet and baselines on MultiWOZ2.2. DSGFNet significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$). 55
4.4 Joint GA of DSGFNet and baselines on MultiWOZ2.1. DSGFNet achieves comparable performance of the best baseline. 56
4.5 Ablation study on unseen domains of SGD, all domains of SGD, MultiWOZ2.2 and MultiWOZ2.1. 56
4.6 Case study of DSGFNet and Seq2Seq-DU on SGD. Slot values are extracted from the dialogue context with the same color. The relation of yellow high light slot pair is predicted as co-reference. The relation of red underline slot pair is predicted as co-update. The relation of bold font slot pair is predicted as co-occurrence. Slot values in red high light are incorrectly predicted ones. 58
4.7 Performance comparison with different dynamic slot relations and fully-connected relations on unseen domains of SGD, all domains of SGD, MultiWOZ2.2 and MultiWOZ2.1. 58
4.8 The proportion of different types of dynamic slot relations on SGD, MultiWOZ2.2, and MultiWOZ2.1 in training sets. . . . 59
4.9 Accuracy of DSGFNet in each domain on SGD test set. Domains marked with * are those for which the schemata in the test set are not present in the training set. Domains marked with ** have both the unseen and seen schemata. For other domains, the schemata in the test set are also seen in the training set. 60

5.1 Number of dialogues, task knowledge, user profiles, and turns in ClariT. 70
5.2 Performance of MAS2S and baselines on automatic metrics. Numbers in **bold** denote best results in that metric. MAS2S significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$). 73
5.3 Average number of clarification questions (NoQ) and absolute difference of clarification questions (AbsDiff) over the dialogues of MAS2S and the best baseline UBAR on ClariT. **Bold** denotes best results. 76
5.4 Performance of MAS2S and MAS2S-w/oProfile on ClariT dataset. Numbers in **bold** denote best results in that metric. MAS2S significantly improves over the MAS2S-w/oProfile (two-sided paired t-test, $p < 0.05$). 77

6.1 Statistics of the task-oriented dialogue datasets. 84
6.2 Performance of SG-USM and baselines on various evaluation benchmarks. Numbers in **bold** denote the best model performance for a given metric. Numbers with * indicate that SG-USM model is better than the best-performing baseline method (underlined scores) with statistical significance (t-test, $p < 0.05$). 84

6.3 Performance of SG-USM and the best baseline USDA on zero-shot learning ability. All the models are trained on SGD and tested on MWOZ and ReDial. Numbers in **bold** denote best results in that metric. Numbers with * indicate that the model is better than the performance of baseline with statistical significance (t-test, p < 0.05). 87

7.1 Sub-tasks schema in CRSAgent. 94
7.2 The details of the prompt design in CRSAgent. There are some injectable slots in the prompts, such as Sub-Task List, Sub-Task Schema, Demonstrations, Dialogue Context, Sub-Task Goal, Expert Models, and Sub-Task Output. These slots are uniformly replaced with the corresponding text before being fed into the LLM. 95
7.3 Statistics of GoRecDial and TG-ReDial in the experiments. . . 99
7.4 Performance of CRSAgent and baselines for recommendation on GoRecDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$). 102
7.5 Performance of CRSAgent and baselines for recommendation on TG-ReDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$). 102
7.6 Performance of CRSAgent and baselines for conversation on GoRecDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline on Distinct (two-sided paired t-test, $p < 0.05$). 102
7.7 Performance of CRSAgent and baselines for conversation on TG-ReDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline on Distinct (two-sided paired t-test, $p < 0.05$). 103
7.8 Effectiveness of the mechanisms to instruct LLM in CRSAgent. Numbers in bold denote the best results. 105

Chapter 1

Introduction

1.1 Task-oriented Dialogue Systems

Task-oriented dialogue systems have gained significant attention in recent years due to their potential to revolutionize human-computer interaction across various domains. These systems aim to facilitate natural language conversations between users and machines to accomplish specific tasks, such as booking a flight, ordering food, or scheduling appointments. The significance of this area lies in its ability to streamline user interactions with technology, making them more intuitive, efficient, and user-friendly.

The task-oriented dialogue systems can effectively improve user experience. Traditional interfaces like forms, menus, and buttons can be cumbersome and unintuitive for many users. Task-oriented dialogue systems offer a more natural and conversational interaction paradigm, which can significantly enhance user experience and satisfaction. By automating routine tasks through conversational interfaces, task-oriented dialogue systems can save time and effort for both users and service providers. This efficiency can lead to increased productivity and cost savings in various domains, such as customer service, e-commerce, and healthcare.

Besides, dialogue systems can also leverage user interactions to personalize responses and recommendations, leading to more tailored and relevant experiences. By understanding user preferences and context, these systems can deliver more accurate and satisfying outcomes.

As the demand for digital services continues to grow, scalable solutions are essential for meeting user needs effectively. Task-oriented dialogue systems can scale relatively easily compared to traditional human-driven processes, making them suitable for handling large volumes of interactions without compromising quality.

Finally, dialogue systems can be integrated across multiple platforms and devices, including smartphones, smart speakers, websites, and messaging apps. This cross-platform compatibility ensures seamless user experiences regardless of the interface or device used.

1.2 Research Questions

In contrast to open-domain dialog systems, which prioritize maximizing user engagement [1], task-oriented dialog systems focus on achieving particular

tasks within one or multiple domains [2]. Typically, these systems are constructed upon structured schema or unstructured documents, which delineate the domain knowledge required for the tasks.

Existing studies on task-oriented dialog systems can be broadly classified into two categories: pipeline and end-to-end methods. In pipeline methods, the entire system is compartmentalized into several modules, including natural language understanding (NLU) [3, 4], natural language generation (NLG) [5, 6], and user satisfaction modeling (USM) [7, 8]. Conversely, end-to-end methods construct the system using a single model [9, 10], which directly processes a natural language context as input and generates a natural language response.

One major type of issues in task-oriented dialog systems is how to integrate knowledge, which includes domain knowledge and common sense knowledge, into the system to improve performance. For domain knowledge, it includes domain schema and domain document. In most previous methods [11, 12, 13, 14], the domain knowledge is pre-defined and highly dependent on the corpus they use. For example, the slots of the restaurant domain contain address area, cuisine type, and price range; the documents of the restaurant domain contain reviews, menus, recipes, and restaurant introductions. These approaches are coupled with domain knowledge and cannot be easily transferred to a new task. With the emergence of large language models, several approaches [15] have been proposed to leverage the common sense knowledge of large language models within task-oriented dialogue systems to mitigate the transfer problem.

Therefore, the research questions we discuss in this thesis include:

1. How to use the domain schema knowledge to track dialogue state to enhance the natural language understanding ability of the task-oriented dialogue system.

2. How to use domain document knowledge to generate system responses to clarify users' needs and user profile.

3. How to model user satisfaction using domain knowledge to better simulate the user and evaluate the task-oriented dialogue system.

4. How to use common scene knowledge in the large language models to empower the end-to-end task-oriented dialogue systems.

1.3 Significance

Integrating knowledge into task-oriented dialogue systems is crucial for enhancing their effectiveness, accuracy, and overall user experience. Knowledge integration enables dialogue systems to leverage external information sources such as databases, ontologies, and APIs to provide more informative and contextually relevant responses to user queries. This integration can be achieved through both pipeline methods, where knowledge is incorporated into specific modules of the dialogue system, and end-to-end methods, where knowledge is integrated into the entire system's learning process.

Here is a breakdown of the significance of integrating knowledge into task-oriented dialogue systems for each module in pipeline methods and end-to-end methods:

1. Natural Language Understanding:

   Incorporating knowledge sources into NLU modules can help improve entity recognition, intent detection, semantic parsing, and user needs prediction. For example, using domain-specific schema or knowledge graphs can provide NLU models with structured information about entities, relationships, and domain-specific terminology, leading to a more accurate understanding of user inputs.

2. Natural Language Generation:

   Knowledge can be integrated into response generation by enriching the response templates or candidate generation strategies with domain-specific information. For example, using domain-specific documents or structured databases to retrieve relevant information for response generation can ensure that generated responses are accurate and contextually relevant.

3. User Satisfaction Modeling:

   Integrating knowledge into user satisfaction modeling of task-oriented dialogue systems is essential for enhancing user satisfaction, engagement, and overall system effectiveness. By leveraging knowledge about the domain, user preferences, and interaction context, dialogue systems can provide more personalized, relevant, and accurate responses, leading to higher levels of user satisfaction and improved user experiences.

4. End-to-End Task-Oriented Dialogue Systems:

   Integrating knowledge into end-to-end task-oriented dialogue systems enhances their capabilities in understanding user inputs, generating contextually relevant responses, and adapting to diverse domains and tasks. This leads to more natural, effective, and satisfying interactions between users and dialogue systems, ultimately improving the overall user experience and usability of the system.

1.4 Contributions

In this thesis, we elaborate on how to integrate knowledge into task-oriented dialog systems which include each module in pipeline methods and end-to-end methods. Our contributions are as follow:

1. Schema-guided natural language understanding.

   Natural language understanding in task-oriented dialogue systems aims to keep track of users' intentions during the course of a conversation. We propose a new approach to dialogue state tracking, referred to as Seq2Seq-DU, which formalizes natural language understanding in task-oriented dialogue as a sequence-to-sequence problem. Seq2Seq-DU employs two BERT-based encoders to respectively encode the utterances in the dialogue and the descriptions of schemas, an attender to calculate attentions between the utterance embeddings and the schema embeddings, and a decoder to generate pointers to represent the current state of dialogue. Seq2Seq-DU has the following advantages. It can

jointly model intents, slots, and slot values; it can leverage the rich representations of utterances and schemas based on BERT; it can effectively deal with categorical and non-categorical slots, and unseen schemas.

2. Relation-guided natural language understanding.

   Natural language understanding in task-oriented dialogue is challenging, although significant progresses have been made recently. In natural language understanding, modeling the relations among domains and slots is still an under-studied problem. Existing approaches that have considered such relations generally fall short in: (1) fusing prior slot-domain membership relations and dialogue-aware dynamic slot relations explicitly, and (2) generalizing to unseen domains. To address these issues, we propose a novel **D**ynamic **S**chema **G**raph **F**usion **Net**work (**DSGFNet**), which generates a dynamic schema graph to explicitly fuse the prior slot-domain membership relations and dialogue-aware dynamic slot relations. It also uses the schemata to facilitate knowledge transfer to new domains. DSGFNet consists of a dialogue utterance encoder, a schema graph encoder, a dialogue-aware schema graph evolving network, and a schema graph enhanced dialogue state decoder.

3. Document-guided natural language generation.

   Natural language generation in task-oriented dialogue systems aims to generate system responses to reply to users' requests. To provide accurate and personalized system responses, task-oriented dialogue systems need to address two potential issues: 1) users' inability to describe their complex information needs in their requests; and 2) ambiguous/missing information the system has about the users. To address these issues, task-oriented dialogue systems must have the ability to proactively ask questions so as to *clarify* necessary information based on task knowledge. We propose a **M**ulti-**A**ttention **S**eq**2S**eq Network, named **MAS2S**, which can generate questions to clarify the user's information needs and the user's profile considering unstructured task knowledge. MAS2S consists of a text encoder, an answer confidence embedding network, and a natural language decoder. In particular, the answer confidence embedding network takes into account the unstructured task knowledge to derive confidence of the predicted answer to the user's request. We further build a dataset, called ClariT, which contains about 100k task-oriented dialogues. Our experimental results on ClariT show that MAS2S outperforms baselines on both clarification question generation and answer prediction.

4. Schema-guided user satisfaction modeling.

   User satisfaction modeling is one of the popular choices for task-oriented dialogue systems evaluation, where user satisfaction typically depends on whether the user's task goals were fulfilled by the system. Task-oriented dialogue systems use task schema, which is a set of task attributes, to encode the user's task goals. Existing studies on user satisfaction modeling neglect explicitly modeling the user's task goals fulfillment using the task schema. We propose **SG-USM**, a novel **S**chema-**G**uided **U**ser **S**atisfaction **M**odeling framework. It explicitly

models the degree to which the user's preferences regarding the task attributes are fulfilled by the system for predicting the user's satisfaction level. SG-USM employs a pre-trained language model for encoding dialogue context and task attributes. Further, it employs a fulfillment representation layer for learning how many task attributes have been fulfilled in the dialogue, an importance predictor component for calculating the importance of task attributes. Finally, it predicts the user satisfaction based on task attribute fulfillment and task attribute importance.

5. Large language model empowered end-to-end method.

   A task-oriented dialog system is designed to assist users in completing specific tasks. The task usually contains multiple sub-tasks. For example, in the conversational recommendation domain, the sub-tasks usually contain user preference elicitation, recommendation, explanation, and item information search. To develop effective task-oriented dialog system, there are some challenges: 1) how to properly manage sub-tasks; 2) how to effectively solve different sub-tasks; and 3) how to correctly generate responses that interact with users. Recently, Large Language Models (LLMs) have exhibited an unprecedented ability to reason and generate with common sense knowledge, presenting a new opportunity to develop more powerful task-oriented dialog systems. To address the challenges listed above, we propose a new LLM-based agent for task-oriented dialog systems, referred to as **CRSAgent**. For sub-task management, we leverage the reasoning ability of LLMs to effectively manage sub-tasks. For sub-task solving, we merge LLMs with expert models of different sub-tasks to achieve an enhanced performance. For response generation, we utilize the generation ability of LLMs as a language interface to better interact with users. Specifically, CRSAgent divides the workflow into four stages: sub-task detection, model matching, sub-task execution, and response generation. CRSAgent also designs schema-based instruction, demonstration-based instruction, dynamic sub-task and model matching, and summary-based generation to instruct LLMs to generate desired results in the workflow. Finally, to adapt LLMs to a specific domain, we also propose to fine-tune LLMs with reinforcement learning from task performance feedback, referred to as RLPF. Experimental results on benchmark datasets show that CRSAgent with RLPF outperforms the existing methods.

1.5 Publication List

The subsequent chapters of the thesis are heavily based on the following published papers.

Chapter 3. Schema-Guided Natural Language Understanding:

Yue Feng, Yang Wang, and Hang Li, A Sequence-to-Sequence Approach to Dialogue State Tracking. Annual Meeting of the Association for Computational Linguistics (ACL), 2021

Chapter 4. Relation-Guided Natural Language Understanding:

Yue Feng, Aldo Lipani, Fanghua Ye, Qiang Zhang, and Emine Yilmaz. Dynamic Schema Graph Fusion Network for Multi-Domain Dialogue State

Tracking. Annual Meeting of the Association for Computational Linguistics (ACL), 2022

Chapter 5. Document-Guided Natural Language Generation:

Yue Feng, Hossein A Rahmani, Aldo Lipani, Emine Yilmaz. Towards Asking Clarification Questions for Information Seeking on Task-Oriented Dialogues. Arxiv, 2023. Under review of COLING 2025.

Chapter 6. Schema-Guided User Satisfaction Modeling:

Yue Feng, Yunlong Jiao, Animesh Prasad, Nikolaos Aletras, Emine Yilmaz, and Gabriella Kazai. Schema-Guided User Satisfaction Modeling for Task-Oriented Dialogues. Annual Meeting of the Association for Computational Linguistics (ACL), 2023

Chapter 7. Large Language Model Empowered End-to-End Task-Oriented Dialogue Systems:

Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, Fei Sun, Emine Yilmaz. A Large Language Model based Agent for Conversational Recommender System. Arxiv, 2023. Under review of TOIS.

1.6 A Readers Guide

This thesis is structured as follows. In Chapter. 2, we introduce the recent related work of each component in pipeline methods and end-to-end approaches. In Chapter. 3, we discuss the schema-guided natural language understanding method. In Chapter. 4, we discuss the relation-guided natural language understanding method. In Chapter. 5, we discuss the document-guided natural language generation method. In Chapter. 6, we discuss the schema-guided user satisfaction modeling method. In Chapter. 7, we discuss the large language model empowered end-to-end method. Finally in Chapter. 8, we conclude the thesis and discuss future research trends.

Chapter 2

Related Work

The architecture of task-oriented dialog systems can be broadly categorized into two main classes: pipeline and end-to-end approaches. In pipeline approaches, the system typically comprises multiple components, including Natural Language Understanding (NLU), Natural Language Generation (NLG), and User Satisfaction Modeling (USM), arranged sequentially as depicted in Figure 2.1. The NLU, NLG, and USM components are often trained independently before being integrated. Notably, while the NLU-NLG-USM framework represents a common configuration of the pipeline system, alternative configurations also exist. Recent research has explored merging certain components, such as word-level dialogue state tracking and word-level policy, leading to diverse pipeline configurations [3, 4, 5, 6].

In end-to-end approaches, dialog systems are trained in an end-to-end manner, without specifying each individual component. Commonly, the training process is formulated as generating a responding utterance given the dialog context and the backend knowledge base.

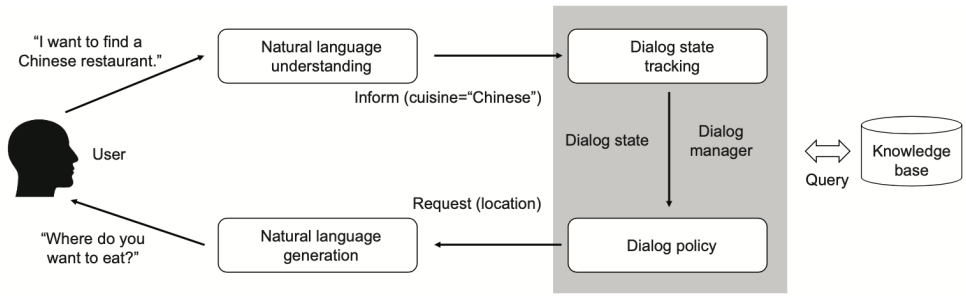


Figure 2.1: General framework of a pipeline task-oriented dialog system.

2.1 Task-oriented Dialogue Systems

Developing task-oriented (also known as goal-oriented) dialogue systems has garnered significant attention within both the research community and the industry. Task-oriented dialogue systems are designed to aid users in completing specific tasks within defined domains, such as restaurant booking, weather inquiries, and flight reservations, rendering them invaluable for real-world applications. Unlike open-domain dialogue systems, which primarily focus on maximizing user engagement [1], task-oriented dialogue systems concentrate on achieving particular tasks within one or multiple domains [2].

Generally, these systems rely on structured ontologies to encapsulate domain knowledge pertinent to the tasks.

Existing research on task-oriented dialogue systems can be broadly categorized into two main approaches: pipeline and end-to-end methods. In pipeline methods, the system is segmented into multiple modules, encompassing natural language understanding (NLU), dialogue state tracking (DST), dialogue policy (Policy), natural language generation (NLG), and user satisfaction modeling (USM). Additionally, there are hybrid approaches like word-level DST [3, 4] (integrating NLU and DST) and word-level policy [5, 6] (integrating Policy and NLG). On the other hand, end-to-end methods construct the system using a single model, which directly processes natural language input and produces natural language responses.

Constructing a pipeline system typically necessitates extensive labeled dialogue data to train each component. The modular architecture renders the system more interpretable and robust compared to end-to-end counterparts. Consequently, many real-world commercial systems adopt this approach. End-to-end systems demand fewer annotations, facilitating easier construction. Nevertheless, their holistic structure renders them akin to black boxes, which can be more challenging to control [7].

In pipeline methods, recent research has placed greater emphasis on the dialogue state tracking and dialogue policy components, collectively referred to as Dialog Management. This is because both the natural language understanding (NLU) and natural language generation (NLG) components represent standalone language processing tasks that are less tightly integrated with other aspects of dialogue systems. Dialogue state tracking (DST) tasks, based on domain ontology, can be viewed as a classification task involving the prediction of each slot's value. However, such classification-based methods may encounter challenges, such as the out-of-vocabulary (OOV) problem and difficulty in generalizing to new domains when training data are insufficient. Dialogue policy learning is often framed as a reinforcement learning (RL) task. However, unlike other well-established RL tasks like video game playing [8] and Go [9], training a dialogue policy requires human interaction, making it resource-intensive. Additionally, many existing methods rely on manually defined rewards, such as task-completion rates and session turn numbers, which may not reliably evaluate system performance.

In end-to-end methods, the inherent demand for abundant data by the standard sequence-to-sequence model poses challenges in mastering the intricate slot filling mechanism within task-oriented dialogue systems, especially when confronted with a restricted corpus of domain-specific data. Additionally, addressing the knowledge base query issue entails the model generating an intermediate query alongside the encoder and decoder, a task that is not inherently straightforward. Another limitation lies in the word-level strategy employed by the encoder-decoder framework, which could potentially result in suboptimal performance due to the entanglement of strategy and language functions.

2.2 Natural Language Understanding for Task-oriented Dialogue Systems

When presented with a user utterance, the natural language understanding (NLU) component interprets it and maps it onto a structured semantic representation. One prevalent schema for such representation is the dialog act, which comprises intent and slot-values, as outlined in Table 2.1. Intent type serves as a high-level categorization of the utterance, distinguishing between functions like Query and Inform, thereby indicating the purpose of the utterance. Slot-value pairs represent task-specific semantic elements mentioned in the utterance. It is important to note that both intent type and slot-value pairs are task-specific, aligning with the ontology and facilitating queries to the knowledge base.

Table 2.1: An example of dialog act for an utterance in the restaurant reservation domain.

Utterance	Intent	Slot value
How about a British restaurant in north part of town	Query	Cuisine= British, Location= North

Utilizing the dialog act structure, the NLU task can be broken down into two distinct tasks: intent detection and slot-value extraction. Intent detection typically involves classifying the utterance to determine its intent, while slot-value extraction is commonly approached as a sequence labeling problem, aiming to recognize and extract relevant slot-values from the utterance.

$$p_{intent}(d|x_1, x_2, ..., x_n), \tag{2.2.1}$$

$$p_{s-v}(y_1, y_2, ..., y_n|x_1, x_2, ..., x_n), \tag{2.2.2}$$

where the d indicates the intent class and y_1 to y_n are the labels of each token in the utterance $[x_1, x_2, ..., x_n]$ in which x_i is a token and n means the number of tokens.

Thanks to the robust capabilities of sequence modeling, RNN and its variants have become prevalent choices in intent detection and slot-value extraction [16, 17, 18]. These models leverage the hidden state of each token to predict the corresponding label y_i, and utilize the final hidden state to discern the sentence intent d. Other neural network architectures, such as recursive neural networks [19] and convolutional neural networks (CNNs) [20], have also been investigated for these tasks. Additionally, conditional random fields, commonly employed in traditional sequence tagging models, have been integrated with RNNs [21] and CNNs [20] to enhance performance. More recently, pre-trained models like BERT [22] have emerged as another popular option [23, 24].

Additionally, some models have focused on strengthening the connection between intent classification and slot tagging. Goo et al. [25] proposed an intent gate to guide the slot tagging process, while Liu et al. [26] proposed an attention mechanism to facilitate the interaction between word and sentence representations.

The dialog state tracker gauges the user's objective at each time step by considering the entire dialog context as input. The dialog state at time t can be seen as a condensed representation of the preceding turns up to t. Early

studies presumed the fixed sets of dialog states and depicted state transitions during interaction as a Markov Decision Process (MDP). POMDP, on the other hand, assumes partial observability of the observation, enhancing robustness in intricate scenarios [27, 28, 29]. More recent approaches have embraced belief states for dialog state representation, where the state comprises slot-value pairs delineating the user's objective. Consequently, this issue can be framed as a multi-task classification task [30, 31, 32]:

$$p_i(d_{i,t}|u_1, u_2, ..., u_t), \tag{2.2.3}$$

where for each specific slot i, there is a tracker p_i. u_t represents the utterance in turn t. The class of slot i in the t-th turn is $d_{i,t}$. However, this approach encounters limitations when confronted with previously unseen values during runtime. Additionally, some works have approached the natural language processing task of a task-oriented dialogue system by formulating it as a reading comprehension task [33, 34].

In more recent methods, slots are often categorized into two types: free-form and fixed vocabulary [35]. The former type lacks a predetermined vocabulary for the slot, making it unsuitable for value prediction through classification. In the case of free-form slots, one approach involves direct value generation [36], or alternatively, predicting the span of the value within the utterance [37]. Generative methods typically employ a decoder to generate slot values word by word from an extensive vocabulary. However, this method may encounter challenges with rare words due to vocabulary limitations. On the other hand, span-based methods assume that the value is presented within the context, enabling the model to predict the start and end positions of that span.

Recent NLU approaches of task-oriented dialogue systems mainly focus on encoding the dialogue contexts with deep neural networks (e.g., convolutional and recurrent networks) and inferring the values of slots independently [38, 39, 4, 36, 40, 41, 42]. With the prevalence of pre-trained language models, such as BERT [43] and GPT-2 [44], a great variety of NLU approaches have been developed on top of these pre-trained models [40, 45, 46]. The relations among domains and slots are not considered in the above approaches. However, the prior slot-domain membership relations can facilitate the sharing of domain knowledge and the dialogue-aware dynamic slot relations can conduce dialogue history understanding. Ignoring these relations may lead to sub-optimal performance.

To fill in this gap, several new NLU approaches, which involve the relations among domains and slots, have been proposed. Some of them leverage a graph structure to capture the slot-domain membership relations [47, 48, 49, 50, 51]. Specifically, a predefined schema graph is employed to represent the slot-domain membership relations. However, they fail to incorporate the dialogue-aware dynamic slot relations into the schema graph. The other approaches utilize the attention mechanism to learn dialogue-aware dynamic slot relation features in order to facilitate information flow among slots [37, 52, 53, 41, 54]. However, these approaches ignore the slot-domain membership relations defined by prior knowledge. Since both the prior slot-domain membership relations and dialogue-aware dynamic slot relations can enhance DST performance, our approach is developed to combine them in an effective way.

Given that a deployed dialogue system may encounter an ever-increasing number of new domains that have limited training data available, the NLU

module should be capable of generalizing to unseen domains. Recent NLU approaches have focused on using zero-shot learning to achieve this goal [55, 56]. These approaches exploit the natural language descriptions of schemata to transfer knowledge across domains. However, they ignore the relations among domains and slots. In this work, we propose a unified framework to fuse the prior slot-domain membership relations and dialogue-aware dynamic slot relations, no matter whether the domains are seen or not.

2.3 Natural Language Generation for Task-Oriented Dialogue Systems

Once the dialog act is generated by the dialog policy, the natural language generation component translates the act into a natural language utterance, often framed as a conditioned language generation task [57]. This task entails taking the dialog act as input and producing the corresponding natural language response. To enhance the user experience, the generated utterance should (1) effectively convey the semantics of the dialog act for task completion, and (2) exhibit naturalness, specificity, and informativeness akin to human language. Another challenge is constructing a robust NLG system with limited training data. Peng et al. [58] proposed SC-GPT, which involves pre-training GPT with a large-scale NLG corpus collected from publicly available dialog datasets, followed by fine-tuning the model on target NLG tasks using only a few training instances.

Traditional NLG approaches for task-oriented dialogues have historically relied on template-based and rule-based methods [59, 60], where experts manually design templates and rules tailored to specific domains. Subsequently, a shift towards data-driven methodologies employing machine learning emerged [61, 62, 63]. Kondadadi et al. [64] introduced a statistical approach for generating utterances using techniques like k-means clustering and support vector machines. More recently, a multitude of generation models based on end-to-end learning have surfaced, leveraging deep learning techniques [65, 66, 67]. Wen et al. [57] proposed SC-LSTM, which regulates utterance generation by incorporating dialogue act (DA) feature vectors and reading gates. SC-GPT [58] stands out as a state-of-the-art model for MultiWOZ, achieving remarkable performance through fine-tuning the language model GPT-2 [44] on extensive task-oriented dialog datasets.

Certain end-to-end models [68, 6] produce system utterances directly from dialogue history, bypassing traditional NLG methods, a strategy known as word-level policy. Notably, Zhao et al. [5] and Mehri et al. [69] refine word-level policy through reinforcement learning (RL) to enhance task completion. However, while these approaches leverage RL for system utterance generation, they do not address the adaptation of the NLG module itself to varying environments and user preferences.

Various methods have emerged for tailoring utterances to suit the user. Walker et al. [70] employed quantitative user modeling in multimodal dialogues, crafting speech output that considers user preferences. Janarthanam and Lemon [71] utilized reinforcement learning (RL) to generate utterances aligned with the user's domain knowledge. Duek and Jurcí cek [72] introduced an NLG approach capable of producing utterances reflecting entrainment.

Additionally, Mairesse and Walker [73] proposed PERSONAGE, an NLG framework adept at generating utterances reflecting the Big Five personality traits. Our study distinguishes itself from these prior works by optimizing an existing NLG to accurately convey dialogue acts (DAs) tailored to specific environments and users.

Over the past few years, numerous methods leveraging reinforcement learning (RL) for language generation tasks have emerged [74]. These approaches span various domains, including machine translation methods [75, 76] that employ BLEU as the reward function, summary generation techniques [77, 78] utilizing ROUGE, and narrative generation [79]. Furthermore, several methodologies have incorporated human feedback rather than relying solely on automatic evaluation metrics, evident in machine translation [80], summary generation [81, 82], and open-domain dialogue [83, 84]. Our study investigates the potential application of these recent advancements in NLG for task-oriented dialogues.

Despite the extensive attention that task-oriented dialogue systems have received, our understanding of the nature of information-seeking in task-oriented dialogues is still limited, and there lacks a task-oriented information-seeking paradigm that can ask clarification questions to users which is one of the goals of this work.

Research on clarification question generation has attracted considerable attention in the fields of NLP and IR in recent years. Zhang et al. [85] proposed to ask aspect-based clarification questions in the right order so as to understand the user's needs. Aliannejadi et al [86] and Xu et al. [87] proposed a ranking model to select clarification questions in open-domain information-seeking conversations. Cao et al. [88] proposed to feed expected question specificity along with the context to generate specific and generic clarifying questions. Rao et al. [89] proposed a sequence-to-sequence generation network using the attention mechanism to generate clarification questions. Kumar et al. [90] generated clarification questions by sampling comments from StackExchange posts.

To the best of our knowledge, no previous work exists on clarification question generation for information seeking on task-oriented dialogues, which is the focus of this chapter.

2.4 User Satisfaction Modeling for Task-Oriented Dialogue Systems

Unlike chitchat dialogue systems that aim at conversing with users without specific goals, task-oriented dialogue systems assist users to accomplish certain tasks [11, 91]. Task-oriented dialogue systems can be divided into module-based methods [12, 92, 93, 53, 48, 4, 94, 26] and end-to-end methods [95, 96, 97, 98, 21]. To measure the effectiveness of task-oriented dialogue systems, evaluation is a crucial part of the development process. Several approaches have been proposed including automatic evaluation metrics [99, 3], human evaluation [95, 25], and user satisfaction modeling [7, 8]. Automatic evaluation metrics, such as BLEU [100], make a strong assumption for dialogue systems, which is that valid responses have significant word overlap with the ground truth responses. However, there is significant diversity in the space of valid

responses to a given context [101]. Human evaluation is considered to reflect the overall performance of the system in a real-world scenario, but it is intrusive, time-intensive, and does not scale [102].

Offline evaluation relying on test sets is commonly employed but has limitations, as it typically assesses a single turn and fails to provide insights into the overall system effectiveness or user satisfaction with the dialogue flow [103]. Introducing simulation-based evaluation can address these shortcomings and serve as a valuable option for large-scale automatic assessment [102]. User simulators, designed to mimic user behavior, serve various purposes, including training dialogue managers in offline environments [102] and evaluating dialogue policies [29]. Eckert et al. [104] introduce the initial statistical user simulator, while Scheffler and Young [105] propose a graph-based model. Georgila et al. [106] utilize a Markov Model, and Cuayáhuitl et al. [107] introduce a hidden Markov model. Schatzmann et al. [29] present an agenda-based user simulator, elegantly representing user states as a stack of essential actions, termed the agenda. Zhang and Balog [103] employ an agenda-based user simulator to assess conversational recommender systems. Recent advancements include neural approaches, particularly sequence-to-sequence models [108, 109]. To our knowledge, no prior study explicitly incorporates user satisfaction into user simulations. Diverging from past efforts, our work pioneers the integration of user satisfaction into user simulations to enhance their human-like quality.

Recently, user satisfaction modeling has been proposed as the main evaluation metric for task-oriented dialogue systems, which can address the issues listed above. In task-oriented dialogue (TOD) systems, users aim to accomplish specific tasks such as booking a hotel or reserving a ticket. Throughout the interaction between the user and the TOD system, user satisfaction levels can fluctuate [110]. The prediction of user satisfaction within the dialogue is termed user satisfaction estimation. Sun et al. [7] delve into the study of user satisfaction estimation in TOD systems and propose a benchmark comprising multiple datasets for this purpose. They identify that the primary cause of user dissatisfaction often stems from the system's inability to accurately comprehend the user's requests or effectively manage their requirements. Kim and Lipani [111] introduce a multi-task framework demonstrating the mutual benefits among user satisfaction estimation, action prediction, and utterance generation tasks through positive transfer learning. Ye et al. [112] conceptualize user satisfaction across dialogue turns as an event sequence and leverage its dynamics to predict user satisfaction at any given turn. Hu et al. [113] utilize ChatGPT as a user satisfaction estimator, employing satisfaction scores as feedback to train a dialogue utterance generation model.

User satisfaction in task-oriented dialogue systems is related to whether or not, or to what degree, the user's task goals are fulfilled by the system. Some researchers study user satisfaction from temporal user behaviors, such as click, pause, etc. [114, 115, 8, 116, 117, 118]. Other related studies view dialogue action recognition as an important preceding step to USM, such as request, inform, etc. [114, 111]. However, sometimes the user behavior or system actions are hidden in the users natural language feedback and the system's natural language response [119]. To cope with this problem, a number of methods are developed from the perspective of sentiment analysis [7, 120, 121] and response quality assessment [122, 123]. However, all existing methods

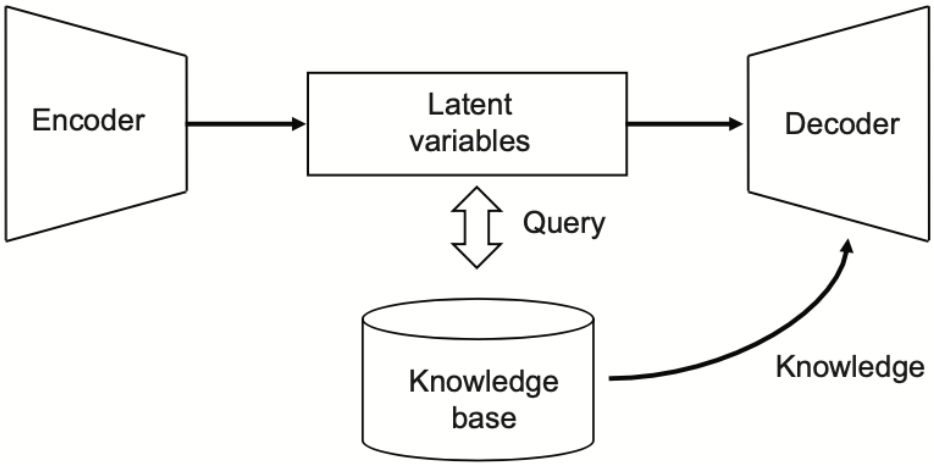


Figure 2.2: Framework of end-to-end dialog systems: Initially, the system encodes the natural language context to derive latent variables, which subsequently aid in the knowledge base (KB) querying. Subsequently, leveraging these latent variables and query results, the decoder generates a natural language response.

cannot explicitly predict user satisfaction with fine-grained explanations, deal with unseen tasks, and alleviate low-resource learning problem. Our work is proposed to solve these issues.

2.5 End-to-End Methods for Task-Oriented Dialogue Systems

In general, the components within a pipeline system are optimized independently. While this modularized structure allows for flexibility, it often results in complex model designs, and the performance of each individual component does not necessarily translate to advancements in the overall system [9]. End-to-end approaches for task-oriented dialog systems draw inspiration from research on open-domain dialog systems, employing neural models to construct the system in a unified, end-to-end manner without modular design, as depicted in Figure 2.2. Many of these methods leverage sequence-to-sequence models as the foundational framework, which is end-to-end differentiable and amenable to optimization using gradient-based methods [124].

In most existing end-to-end approaches, models are typically trained to maximize the prediction probability of responses in collected data. Wen et al. [10] introduced a modularized end-to-end model where each component is represented using neural networks, rendering the model end-to-end differentiable. Bordes et al. [125] formulated task-oriented dialog as a reading comprehension task, treating the dialog history as context, the user utterance as question, and the system response as answer. They employed the end-to-end memory networks for multi-turn inference. Similarly, Madotto et al. [98] adopted a comparable strategy and additionally incorporated knowledge base information into the memory networks. Eric et al. [126] proposed a novel memory network structure called key-value memory networks to extract relevant information from the knowledge base through key-value retrieval. Lei

et al. [94] presented a two-step seq2seq generation model that circumvented structured dialog act representation, retaining only the dialog state representation. In their approach, the model first encodes the dialog history and then generates a dialog state using LSTM and CopyNet. Subsequently, given the state, the model generates the final natural language response.

One major drawback of the aforementioned methods is their reliance on large amounts of training data, which can be costly to acquire. Additionally, they often struggle to fully explore the state-action space since the model's observations are limited to the examples in the data. To address these challenges, reinforcement learning methods have been introduced [127, 128, 129]. Zhao et al. [130] proposed an end-to-end model, where the natural language utterance serves as input, and the system dialog act is generated as a response. In this approach, there is no explicit state representation; instead, the dialog history is encoded into a state vector using LSTM, and DQN is employed to select an action. Williams et al. [131] introduced LSTM-based hybrid code networks (HCN), which support self-defined software.

Two main types of end-to-end TOD have been studied: attribute-based TOD and generation-based TOD.

Attribute-based TOD interact with users through pre-defined actions [132, 133]. They capture user preferences by asking queries about task attributes and generate responses using pre-defined templates. Lei et al. [134] introduced a unified framework aimed at addressing both the determination of relevant questions regarding task attributes and the timing of resluts, leveraging factorization machine estimation. Ren et al. [135] proposed to ask questions by intricately modeling a user's preferences, achieved through the identification of the most pertinent relations from a structured knowledge graph. Xu et al. [136] proposed individual adaptations for the original user embedding and item-level feedback, aligning both with the online attribute-level feedback. Zhang et al. [85] proposed to ask aspect-based questions in a strategic order to comprehend user needs. Simultaneously, personalized search takes place during the conversation, and results are presented when the system attains a sufficient level of confidence. Most of these approaches systematically reduce the hypothesis space to efficiently locate suitable items with fewer iterations. However, this type of TOD often overlooks the importance of generating human-like responses in natural language, potentially impacting user experiences negatively.

Compared to attribute-based TOD, generation-based TOD interact with users through more free-form natural language conversations [137, 138]. They capture the preferences from the conversation context and then generate responses in free-form [139, 140, 141]. Chen et al. [139] proposed to integrate the task model and the dialog generation system by incorporating knowledge-grounded information into users' preferences. Zhou et al. [140] incorporated both word-oriented and entity-oriented knowledge graphs to enrich data representations to align the word-level and entity-level semantic spaces. Tu et al. [142] proposed to model the hierarchical relationships connected by similar users, aiming to provide more comprehensive information to generate responses. Zhou et al. [143] proposed to extract and represent multi-grained semantic units from various data signals, aligning the associated multi-type semantic units in a coarse-to-fine manner. This type of TOD focuses mainly on giving accurate task results through a natural language interface. However,

due to the intricate nature of TOD, their effectiveness is limited. In this study, we focus on the type of generation-based TOD. Our objective is to decompose intricate generation-based TOD into discrete sub-tasks, thereby enhancing performance for each individual sub-task.

With the emergence of LLMs in natural language processing, there has been a growing interest in harnessing the power of LLMs to enhance dialogue systems [144, 145]. For the TOD, there are only some preliminary approaches. Wang et al. [146] proposed to utilize in-context learning to rerank the results of the task model and utilize prompting strategies to generate the responses. However, they ignore the decomposition problem in the TOD. In this work, we target using LLM-based agent to decompose TOD into sub-tasks.

2.6 Clarifying Question Generation and Selection

The problem of query ambiguity is a key motivation for advancing task-oriented dialogue systems over traditional single-turn search. An ambiguous query typically refers to a query where the system cannot confidently discern the users information need, resulting in less relevant responses [147]. Ambiguity can arise from various factors, such as multiple distinct interpretations, under-specified subtopics [148], anaphoric references, or syntactic complexities [149].

Efforts to address query ambiguity generally fall into two categories: 1) Query Suggestion: Offering related queries to guide the user, as explored in [150, 151, 152]. 2) Asking Clarifying Questions: Engaging users proactively to provide additional context, as discussed in [153, 154, 89, 155].

While these approaches share structural and functional similarities, they are not interchangeable. Each has strengths suited to specific scenarios. For instance, clarifying questions are particularly useful for resolving ambiguous queries lacking context, whereas query reformulation is more effective in context-rich situations. Query suggestion, on the other hand, excels in guiding task completion and uncovering user needs.

Among the approaches to resolving query ambiguity, asking clarifying questions (CQs) is the most extensively studied [147] and is often regarded as more convenient due to its proactive nature [156, 157]. Existing research on CQs can be broadly categorized into two main approaches: (1) ranking/selecting CQs, as explored in [158, 153], and (2) generating CQs, as demonstrated in works like [159, 89].

Rao and Daumé III [89] employed generative adversarial learning to train a sequence-to-sequence model for question generation. Zamani et al. [155] introduced a rule-based template completion model alongside two neural question generation models that generate CQs based on the query and its aspects. Subsequent studies [160] highlighted the effectiveness of using templates to guide CQ generation, effectively framing the CQ generation problem as a selection task. Similarly, Sekuli et al. [161] proposed a query facet-driven approach to CQ generation. More recently, Zhao et al. [162] demonstrated that query facets can be extracted from web search results to guide the question generation process, further enhancing the relevance and specificity of CQs.

Chapter 3

Schema-Guided Natural Language Understanding for Task-Oriented Dialogues

This work is concerned with natural language understanding in a task-oriented dialogue system. Building a natural language understanding module that is highly effective is still a challenging issue, although significant progresses have been made recently. This work proposes a new approach to dialogue state tracking, referred to as Seq2Seq-DU, which formalizes natural language understanding in a task-oriented dialogue system as a sequence-to-sequence problem. Seq2Seq-DU employs two BERT-based encoders to respectively encode the utterances in the dialogue and the descriptions of schemas, an attender to calculate attentions between the utterance embeddings and the schema embeddings, and a decoder to generate pointers to represent the current state of dialogue. Seq2Seq-DU has the following advantages. It can jointly model intents, slots, and slot values; it can leverage the rich representations of utterances and schemas based on BERT; it can effectively deal with categorical and non-categorical slots, and unseen schemas. Experimental results on benchmark datasets in different settings (SGD, MultiWOZ2.2, MultiWOZ2.1, WOZ2.0, DSTC2, M2M, SNIPS, and ATIS) show that Seq2Seq-DU outperforms the existing methods.

3.1 Introduction

In natural language understanding, a semantic frame representing the content of user utterance is created at each turn of dialogue. Several semantic frames representing the 'states' of dialogue are created and updated in multiple turns of dialogue, which is called dialogue state tracking (DST). Domain knowledge in dialogues is represented by a representation referred to as schema, which consists of possible intents, slots, and slot values. Slot values can be in a pre-defined set, with the corresponding slot being referred to as categorical slot, and they can also be from an open set, with the corresponding slot being referred to as non-categorical slot. Figure 3.1 shows an example of DST.

We think that a DST module should have the following abilities. (1) Global, the model can jointly represent intents, slots, and slot values. (2) Representable, it has strong capability to represent knowledge for the task, on top of a pre-trained language model like BERT. (3) Scalable, the model can deal with

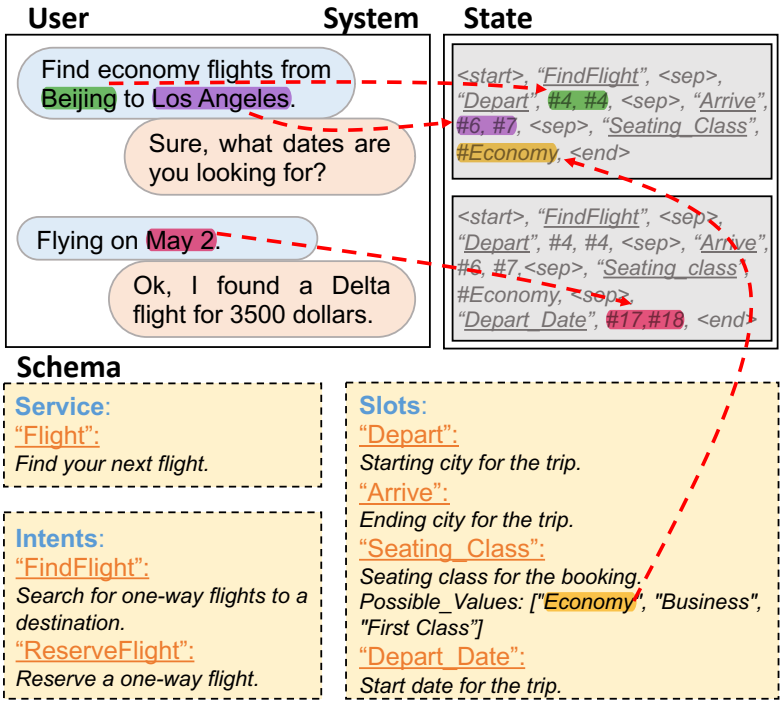


Figure 3.1: An example of dialogue state tracking. Given a dialogue history that contains user utterances and system utterances, and descriptions of schema that contain all possible intents and slot-value pairs, a dialogue state for the current turn is created which is represented by intents and slot-value pairs. There are slot values obtained from the schema (categorical) as well as slot values extracted from the utterances (non-categorical). #4, #6, etc denote pointers.

categorical and non-categorical slots and unseen schemas.

Many methods have been proposed for DST [4, 38, 163, 25]. There are two lines of relevant research. (1) To enhance the scalability of DST, a problem formulation, referred to as schema-guided dialogue, is proposed. In the setting, it is assumed that descriptions on schemas in natural language across multiple domains are given and utilized. Consequently, a number of methods are developed to make use of schema descriptions to increase the scalability of DST [55, 164, 56]. The methods regard DST as a classification and/or an extraction problem and independently infer the intent and slot value pairs for the current turn. Therefore, the proposed models are generally representable and scalable, but not global. (2) There are also a few methods which view DST as a sequence to sequence problem. Some methods sequentially infer the intent and slot value pairs for the current turn on the basis of dialogue history and usually employ a hierarchical structure (not based on BERT) for the inference [94, 36, 165]. Recently, a new approach is proposed which formalizes the tasks in dialogue as sequence prediction problems using a unified language model (based on GPT-2) [166]. The method cannot deal with unseen schemas and intents, however, and thus is not scalable.

We propose a novel approach to DST, referred to as Seq2Seq-DU (sequence-to-sequence for dialogue understanding), which combines the advantages of the existing approaches. To the best of our knowledge, there was no previous work which studied the approach. We think that DST should be formalized as a sequence to sequence or 'translation' problem in which the utterances in the

dialogue are transformed into semantic frames. In this way, the intents, slots, and slot values can be jointly modeled. Moreover, NLU can also be viewed as a special case of DST and thus Seq2Seq-DU can also be applied to NLU. We note that very recently the effectiveness of the sequence to sequence approach has also been verified in other language understanding tasks [167].

Seq2Seq-DU comprises a BERT-based encoder to encode the utterances in the dialogue, a BERT based encoder to encode the schema descriptions, an attender to calculate attentions between the utterance embeddings and schema embeddings, and a decoder to generate pointers of items representing the intents and slots-value pairs of state.

Seq2Seq-DU has the following advantages. (1) Global: it relies on the sequence to sequence framework to simultaneously model the intents, slots, and slot-values. (2) Representable: It employs BERT [43] to learn and utilize better representations of not only the current utterance but also the previous utterances in the dialogue. If schema descriptions are available, it also employs BERT for the learning and utilization of their representations. (3) Scalable: It uses the pointer generation mechanism, as in the Pointer Network [168], to create representations of intents, slots, and slot-values, no matter whether the slots are categorical or non-categorical, and whether the schemas are unseen or not.

Experimental results on benchmark datasets show that Seq2Seq-DU[1] performs much better than the baselines on SGD, MultiWOZ2.2, and MultiWOZ2.1 in multi-turn dialogue with schema descriptions, is superior to BERT-DST on WOZ2.0, DSTC2, and M2M, in multi-turn dialogue without schema descriptions, and works equally well as Joint BERT on ATIS and SNIPS in single turn dialogue (in fact, it degenerates to Joint BERT).

3.2 Seq-to-Seq Dialogue State Tracking

Our approach Seq2Seq-DU formalizes dialogue state tracking as a sequence to sequence problem using BERT and pointer generation. As shown in Figure 3.2, Seq2Seq-DU consists of an utterance encoder, a schema encoder, an utterance schema attender, and a state decoder. In each turn of dialogue, the utterance encoder transforms the current user utterance and the previous utterances in the dialogue into a sequence of utterance embeddings using BERT; the schema encoder transforms the schema descriptions into a set of schema embeddings also using BERT; the utterance schema attender calculates attentions between the utterance embeddings and the schema embeddings to create attended utterance and schema representations; finally, the state decoder sequentially generates a state representation on the basis of the attended representations using LSTM and pointer generation.

3.2.1 Utterance Encoder

The utterance encoder takes the current user utterance as well as the previous utterances (user and system utterances) in the dialogue (a sequence of tokens) as input and employs BERT to construct a sequence of utterance embeddings.

[1]The code is available at `https://github.com/sweetalyssum/Seq2Seq-DU`.

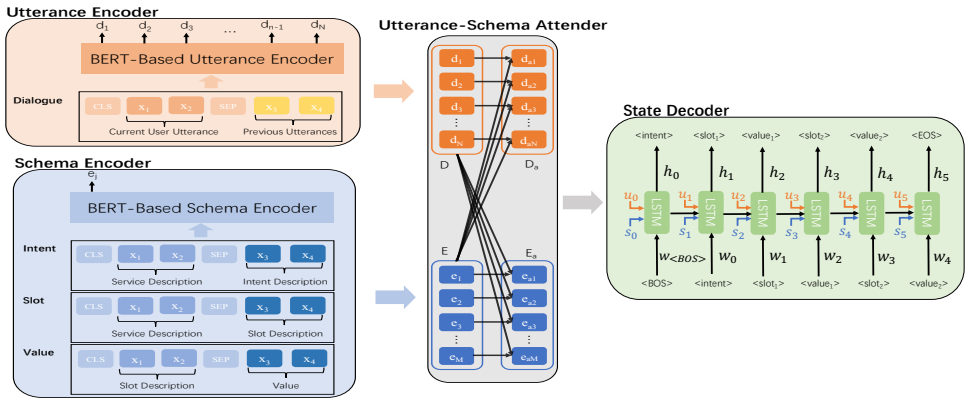


Figure 3.2: The architecture of Seq2Seq-DU, containing utterance encoder, schema encoder, utterance-schema attender, and state decoder.

The relations between the current utterance and the previous utterances are captured by the encoder.

The input of the encoder is a sequence of tokens with length N, denoted as $X = (x_1, ..., x_N)$. The first token x_1 is [CLS], followed by the tokens of the current user utterance and the tokens of the previous utterances, separated by [SEP]. The output is a sequence of embeddings also with length N, denoted as $D = (d_1, ..., d_N)$ and referred to as utterance embeddings, with one embedding for each token.

3.2.2 Schema Encoder

The schema encoder takes the descriptions of intents, slots, and categorical slot values (a set of combined sequences of tokens) as input and employs BERT to construct a set of schema embeddings.

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

Table 3.1: Descriptions for a dialogue schema. Two combined descriptions are used for describing an intent, a slot, or a value in the schema.

Suppose that there are I intents, S slots, and V categorical slot values in the schemas. Each schema element is described by two descriptions as outlined in Table 3.1. The input is a set of combined sequences of tokens, denoted as $Y = \{y_1, ..., y_M\}$. Note that $M = I + S + V$. Each combined sequence starts with [CLS], followed by the tokens of the two descriptions with [SEP] as a separator. The final representation of [CLS] is used as the embedding of the input intent, slot, or slot value. The output is a set of embeddings, and all the embeddings are called schema embeddings $E = \{e_1, ..., e_M\}$.

The schema encoder in fact adopts the same approach of schema encoding as in [55]. There are two advantages with the approach. First, the encoder can be trained across different domains. Schema descriptions in different domains

can be utilized together. Second, once the encoder is fine-tuned, it can be used to process unseen schemas with new intents, slots, and slot values.

3.2.3 Utterance-Schema Attender

The utterance-schema attender takes the sequence of utterance embeddings and the set of schema embeddings as input and calculates schema-attended utterance representations and utterance-attended schema representations. In this way, information from the utterances and information from the schemas are fused.

First, the attender constructs an attention matrix, indicating the similarities between utterance embeddings and schema embeddings. Given the i-th utterance token embedding d_i and j-th schema embedding e_j, it calculates the similarity as follows,

$$A(i,j) = r^\intercal \text{tanh}(W_1 d_i + W_2 e_j), \tag{3.2.1}$$

where r, W_1, W_2 are trainable parameters.

The attender then normalizes each row of matrix A as a probability distribution, to obtain matrix $\overline{A}$. Each row represents the attention weights of schema elements with respect to an utterance token. Then the schema-attended utterance representations are calculated as $D_a = E\overline{A}^\intercal$. The attender also normalizes each column of matrix A as a probability distribution, to obtain matrix $\widetilde{A}$. Each column represents the attention weights of utterance tokens with respect to a schema element. Then the utterance-attended schema representations are calculated as $E_a = D\widetilde{A}$.

3.2.4 State Decoder

The state decoder sequentially generates a state representation (semantic frame) for the current turn, which is represented as a sequence of pointers to elements of the schemas and tokens of the utterances (cf., Figure 7.1). The sequence can then be either re-formalized as a semantic frame in dialogue state tracking[2],

$$[intent; (slot_1, value_1); (slot_2, value_2); ...],$$

or a sequence of labels in NLU (intent-labeling and slot-filling). The pointers point to the elements of intents, slots, and slot values in the schema descriptions (categorical slot values), as well as the tokens in the utterances (non-categorical slot values). The elements in the schemas can be either words or phrases, and the tokens in the utterances form spans for extraction of slot values.

The state decoder is an LSTM using pointer [168] and attention [169]. It takes the two representations D_a and E_a as input. At each decode step t, the decoder receives the embedding of the previous item w_{t-1}, the utterance context vector u_t, the schema context vector s_t, and the previous hidden state h_{t-1}, and produces the current hidden state h_t:

$$h_t = \text{LSTM}(w_{t-1}, h_{t-1}, u_t, s_t). \tag{3.2.2}$$

[2]For simplicity, we assume here that there is only one semantic frame in each turn. In principle, there can be multiple frames.

We adopt the attention function in [169] to calculate the context vectors as follows,

$$u_t = \text{attend}(h_{t-1}, D_a, D_a), \tag{3.2.3}$$
$$s_t = \text{attend}(h_{t-1}, E_a, E_a). \tag{3.2.4}$$

The decoder then generates a pointer from the set of pointers in the schema elements and the tokens of the utterances on the basis of the hidden state h_t. Specifically, it generates a pointer of item w according to the following distribution,

$$z_w = q^\intercal \tanh(U_1 h_t + U_2 k_w), \tag{3.2.5}$$
$$P(\#w) = \text{softmax}(z_w), \tag{3.2.6}$$

where $\#w$ is the pointer of item w, k_w is the representation of item w either in the utterance representations D_a or in the schema representations E_a, q, U_1, and U_2 are trainable parameters, and softmax is calculated over all possible pointers.

During decoding, the decoder employs beam search to find the best sequences of pointers in terms of probability of sequence.

3.2.5 Training

The training of Seq2Seq-DU follows the standard procedure of sequence-to-sequence. The only difference is that it is always conditioned on the schema descriptions. Each instance in training consists of the current utterance and the previous utterances, and the state representation (sequence of pointers) for the current turn. Two pre-trained BERT models are used for representations of utterances and schema descriptions respectively. The BERT models are then fine-tuned in the training process. Cross-entropy loss is utilized to measure the loss of generating a sequence.

3.3 Experiments

3.3.1 Datasets

We conduct experiments using the benchmark datasets on task-oriented dialogue. SGD [55] and MultiWOZ2.2 [164] are datasets for DST; they include schemas with categorical slots and non-categorical slots in multiple domains and natural language descriptions on the schemas, as shown in Table 3.1. In particular, SGD includes unseen schemas in the test set. MultiWOZ2.1 [91] is the previous version of MultiWOZ2.2, which only has categorical slots in multiple domains. WOZ2.0 [170] and DSTC2 [171] are datasets for DST; they contain schemas with only categorical slots in a single domain. M2M [172] is a dataset for DST and it has span annotations for slot values in multiple domains. ATIS [173] and SNIPS [174] are datasets for NLU in single-turn dialogues in a single domain. Table 6.1 gives the statics of datasets in the experiments.

Characteristics	SGD	MultiWOZ2.2	MultiWOZ2.1	WOZ2.0	DSTC2	M2M	ATIS	SNIPS
No. of domains	16	8	7	1	1	2	-	-
No. of dialogues	16,142	8,438	8438	1,612	600	1,500	4,478	13,084
Total no. of turns	329,964	113,556	113,556	23,354	4,472	14,796	4,478	13,084
Avg. turns per dialogue	20.44	13.46	13.46	14.49	7.45	9.86	1	1
Avg. tokens per turn	9.75	13.13	13.38	8.54	11.24	8.24	11.28	9.09
No. of categorical slots	53	21	37	3	3	0	0	0
No. of non-categorical slots	162	40	0	0	0	14	120	72
Have schema description	Yes	Yes	Yes	No	No	No	No	No
Have unseen schemas in test set	Yes	No	No	No	No	No	No	No

Table 3.2: Statistics of datasets in experiments. Numbers are those of training datasets.

3.3.2 Baselines and Variants

We make comparison between our approach and the state-of-the-art methods on the datasets.

SGD, MultiWOZ2.2 and MultiWOZ2.1: We compare Seq2SeqDU with six state-of-the-art methods on SGD, MultiWOZ2.2 and MultiWOZ2.1, which utilize schema descriptions, span-based and candidate-based methods, unified seq2seq model and BERT: FastSGT [56], SGDbaseline [55], TripPy [53], SimpleTOD [166], TRADE [4], and DS-DST [40]. We implemented the baselines using their available open-source code.

WOZ2.0 and DSTC2: Our approach is compared against the state-of-the-art methods on WOZ2.0 and DSTC2, including those using a hierarchical seq2seq model and BERT: COMER [36], BERT-DST [175], StateNet [176], GLAD [38], Belief Tracking [39], and Neural Belief Tracker [163]. Except for Belief Tracking and Neural Belief Tracker, which does not provide open-source code for reproduction, we implemented the baselines using their available open-source code. For elief Tracking and Neural Belief Tracke, we directly use the results reported by the authors.

M2M: We evaluate our approach and the state-of-the-art methods on M2M, which respectively employ a BERT-based architecture and a jointly-trained language understanding model, BERT-DST [175] and DST+LU [177]. We implemented the baselines using their available open-source code.

ATIS and SNIPS: We make comparison between our approach and the state-of-the-art methods on ATIS and SNIPS for NLU within the sequence labeling framework, including Joint BERT [23], Slot-Gated [25], Atten.-BiRNN [26], and RNN-LSTM [18]. We implemented the baselines using their available open-source code.

We also include two variants of Seq2Seq-DU. The differences are whether to use the schema descriptions, and the formation of dialogue state.

Seq2Seq-DU-w/oSchema: It is used for datasets that do not have schema descriptions. It only contains utterance encoder and state decoder.

Seq2Seq-DU-SeqLabel: It is used for NLU in a single-turn dialogue. It views the problem as sequence labeling, and only contains the utterance encoder and state decoder.

3.3.3 Evaluation Measures

We make use of the following metrics in evaluation.

Intent Accuracy: percentage of turns in dialogue for which the intent is correctly identified.

Joint Goal Accuracy: percentage of turns for which all the slots are correctly identified. For non-categorical slots, a fuzzy matching score is used on SGD and exact match are used on the other datasets to keep the numbers comparable with other works.
Slot F1: F1 score to evaluate accuracy of slot sequence labeling.
Statistical Significance Test: A paired t-test is conducted to determine whether there is a significant difference between our method and the best baseline. On the SGD dataset, we compare our method with FastSGD. For MultiWOZ2.1 and 2.2, we compare our method with TripPy. On WOZ2.0, we compare it with COMER. For DSTC2 and M2M, our method is compared with BERT-DST. On the ATIS and SNIP datasets, our method is compared with Joint BERT. The significance level (p-value) is set at 0.05.

3.3.4 Training

We use the pre-trained BERT model ([BERT-Base, Uncased]), which has 12 hidden layers of 768 units and 12 self-attention heads to encode utterances and schema descriptions. The hidden size of LSTM decoder is also 768. The dropout probability is 0.1. We also use beam search for decoding, with a beam size of 5. The batch size is set to 8. Adam [178] is used for optimization with an initial learning rate of 1e-4. Hyper parameters are chosen using the validation dataset in all cases. The training curves of all models are shown in Figure 3.3 and Figure 3.4.

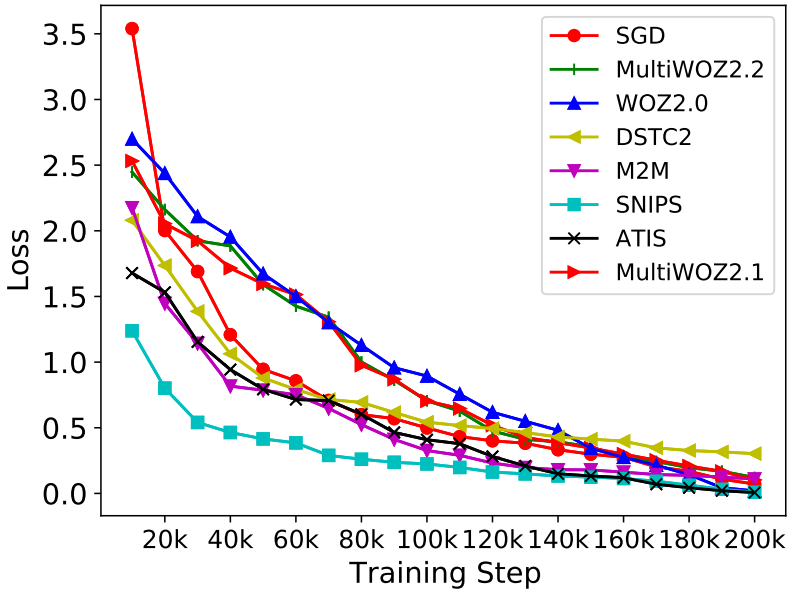


Figure 3.3: Training losses of Seq2Seq-DU on all training datasets.

3.4 Results and Discussion

3.4.1 Experimental Results

Tables 3.3, 3.4, 3.5, and 3.6 show the results. One can see that Seq2Seq-DU performs significantly better than the baselines in DST and performs equally well as the baselines in NLU.

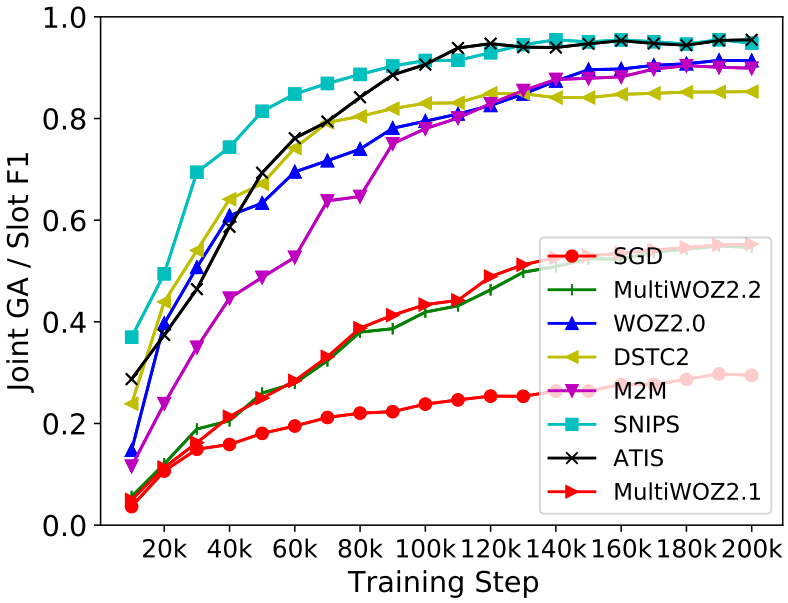


Figure 3.4: Accuracies of Seq2Seq-DU in terms of Join GA / Slot F1 on all test sets.

DST is carried out in different settings in SGD, MultiWOZ2.2, MultiWOZ2.1, WOZ2.0, DSTC2, and M2M. In all cases, Seq2Seq-DU works significantly better than the baselines. The results indicate that Seq2Seq-DU is really a general and effective model for DST, which can be applied to multiple settings. Specifically, Seq2Seq-DU can leverage the schema descriptions for DST when they are available (SGD and MultiWOZ2.2, MultiWOZ2.1)[3]. It can work well in zero-shot learning to deal with unseen schemas (SGD). It can also effectively handle categorical slots (MultiWOZ2.1, WOZ2.0 and DSTC2) and non-categorical slots (M2M). It appears that the success of Seq2Seq-DU is due to its suitable architecture design with a sequence-to-sequence framework, BERT-based encoders, utterance-schema attender, and pointer generation decoder.

NLU is formalized as sequence labeling in ATIS and SNIPS. Seq2Seq-DU is degenerated to Seq2Seq-DU-SeqLabel, which is equivalent to the baseline of Joint Bert. The results suggest that it is the case. Specially, the performances of Seq2Seq-DU are comparable with Joint BERT, indicating that Seq2Seq-DU can also be employed in NLU.

3.4.2 Ablation Study

We also conduct ablation study on Seq2Seq-DU. We validate the effects of three factors: BERT-based encoder, utterance-schema attention, and pointer generation decoder. The results indicate that all the components of Seq2Seq-DU are indispensable.

Effect of BERT

To investigate the effectiveness of using BERT in the utterance encoder and schema encoder, we replace BERT with Bi-directional LSTM and run the

[3]There are better performing systems in the SGD competition. The systems are not based on single methods and thus are not directly comparable with our method.

Model	SGD		MultiWOZ2.2		MultiWOZ2.1	
	Joint GA	Int Acc	Joint GA	Int Acc	Joint GA	Int Acc
SGD-baseline	0.254	0.906	0.420	-	0.434	-
TRADE	-	-	0.454	-	0.460	-
DS-DST	-	-	0.517	-	0.512	-
FastSGT	0.292	0.903	-	-	-	-
SimpleTOD	-	-	-	-	0.514	-
TripPy	-	-	0.535	-	0.553	-
Seq2Seq-DU	**0.301**	**0.910**	**0.544**	**0.909**	**0.561**	**0.911**

Table 3.3: Accuracies of Seq2Seq-DU and baselines on SGD, MultiWOZ2.2 and MultiWOZ2.1 datasets. Seq2Seq-DU outperforms baselines in terms of all metrics.

Model	WOZ2.0	DSTC2
	Joint GA	Joint GA
Neural Belief Tracker	0.842	0.734
Belief Tracking	0.855	-
GLAD	0.881	0.745
StateNet	0.889	0.755
BERT-DST	0.877	0.693
COMER	0.886	-
Seq2Seq-DU-w/oSchema	**0.912**	**0.850**

Table 3.4: Accuracies of Seq2Seq-DU and baselines on WOZ2.0 and DSTC2 datasets. Seq2Seq-DU-w/oSchema performs significantly better than the baselines.

Model	M2M	
	Joint GA	Int Acc
DST+LU	0.767	-
BERT-DST	0.869	-
Seq2Seq-DU-w/oSchema	**0.909**	**0.997**

Table 3.5: Accuracies of Seq2Seq-DU and baselines on M2M dataset. Seq2Seq-DU-w/oSchema significantly outperforms the baselines.

Model	ATIS		SNIPS	
	Slot F1	Int Acc	Slot F1	Int Acc
RNN-LSTM	0.943	0.926	0.873	0.969
Atten.-BiRNN	0.942	0.911	0.878	0.967
Slot-Gated	0.952	0.941	0.888	0.970
Joint BERT	**0.961**	**0.975**	**0.970**	0.986
Seq2Seq-DU-SeqLabel	0.955	0.968	0.965	**0.990**

Table 3.6: Accuracies of Seq2Seq-DU and baselines on ATIS and SNIPS datasets. Seq2Seq-DU-SeqLabel performs comparably with Joint BERT.

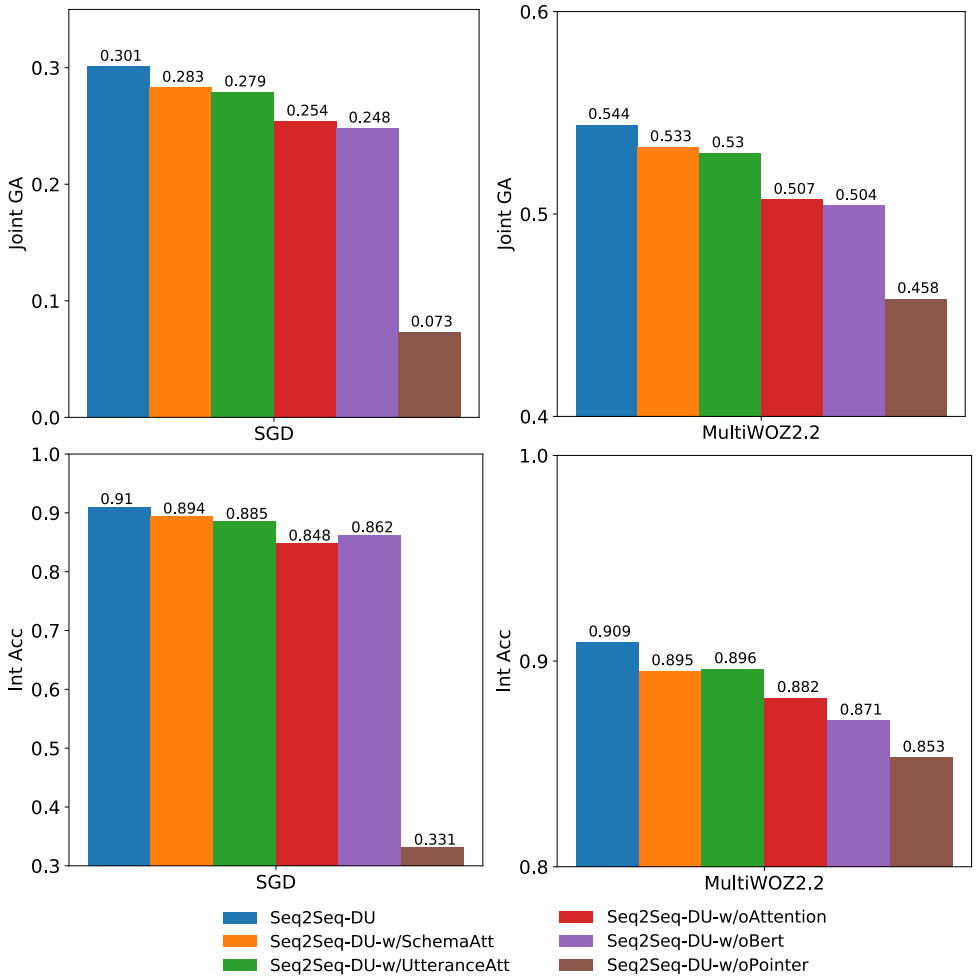


Figure 3.5: Ablation study results of Seq2Seq-DU with respect to BERT, attention, and pointer generation on SGD and MultiWOZ2.2.

model on SGD and MultiWOZ2.2. As shown in Figure 3.5, the performance of the BiLSTM-based model Seq2Seq-DU-w/oBert in terms of Joint GA and Int. Acc decreases significantly compared with Seq2Seq-DU. It indicates that the BERT-based encoders can create and utilize more accurate representations for dialogue understanding.

Effect of Attention

To investigate the effectiveness of using attention, we compare Seq2Seq-DU with Seq2Seq-DU-w/oAttention which eliminates the attention mechanism, Seq2Seq-DU-w/SchemaAtt which only contains the utterance-attended schema representations, and Seq2Seq-DU-w/UtteranceAtt which only contains the schema-attended utterance representations. Figure 3.5 shows the results on SGD and MultiWOZ2.2 in terms of Joint GA and Int. Acc. One can observe that without attention the performances deteriorate considerably. In addition, the performances of unidirectional attentions are inferior to the performance of bidirectional attention. Thus, utilization of bidirectional attention between utterances and schema descriptions is desriable.

Effect of Pointer Generation

To investigate the effectiveness of the pointer generation mechanism, we directly generate words from the vocabulary instead of generating pointers in the decoding process. Figure 3.5 also shows the results of Seq2Seq-DU-w/oPointer on SGD and MultiWOZ2.2 in terms of Joint GA and Int. Acc. From the results we can see that pointer generation is crucial for coping with unseen schemas. In SGD which contains a large number of unseen schemas in the test set, there is significant performance degradation without pointer generation. The results on MultiWOZ2.2, which does not have unseen schemas in the test set, show pointer generation can also make significant improvement on already seen schemas by making full use of schema descriptions.

ID	Dialogue Utterance	Dialogue State	State Predictions of SGD-baseline	State Predictions of Seq2Seq-DU
1	*User*: I wanna rent a place in Campbell. *Sys*: How many baths? *User*: One bath is fine. *Sys*: How many bedrooms? *User*: One bedroom is fine. It also needs in-unit laundry.	"area": Campbell; "in-unit-laundry": True; "intent": rent; "number-of-baths": 1; "number-of-beds": 1; "active-intent": FindHomeByArea;	"area": Campbell; "in-unit-laundry": – ; "intent": rent; "number-of-baths": 1; "number-of-beds": 1; "active-intent": FindHomeByArea;	"area": Campbell; "in-unit-laundry": True; "intent": rent; "number-of-baths": 1; "number-of-beds": 1; "active-intent": FindHomeByArea;
2	*User*: The location isn't really important. It does need to be cheap though, and preferably a guesthouse.	"hotel-area": dontcare ; "hotel-pricerange": cheap; " hotel-type": guesthouse; "active intent": find-hotel;	"hotel-area": dontcare ; "hotel-pricerange": cheap; "hotel-type": hotel; "active intent": find-hotel;	"hotel-area": dontcare; "hotel-pricerange": cheap; "hotel-type": guesthouse; "active intent": find-hotel;

Table 3.7: Case study on Seq2Seq-DU and SGD-baseline on SGD and MultiWOZ2.2. The first example is from SGD and the second is from MultiWOZ2.2. The underlined slot-value pairs represent the ground truth states. The slot-value pairs in blue are correctly predicted ones, while the slot-value pairs in red are incorrectly predicted ones.

3.4.3 Discussions

Case Study

We make qualitative analysis on the results of Seq2Seq-DU and SGD-baseline on SGD and MultiWOZ2.2. We find that Seq2Seq-DU can make more accurate inference of dialogue states by leveraging the relations existing in the utterances and schema descriptions. For example, in the first case in Table 4.6, the user wants to find a cheap guesthouse. Seq2Seq-DU can correctly infer that the hotel type is "guesthouse by referring to the relation between "hotel-pricerange" and "hotel-type". In the second case, the user wants to rent a room with in-unit laundry. In the dataset, a user who intends to rent a room will care more about the laundry property. Seq2Seq-DU can effectively extract the relation between "intent" and "in-unit-laundry", yielding a correct result. In contrast, SGD-baseline does not model the relations in the schemas, and thus it cannot properly infer the values of "hotel-type" and "in-unit-laundry".

Dealing with Unseen Schemas

We analyze the zero-shot learning ability of Seq2Seq-DU. Table 4.9 presents the accuracies of Seq2Seq-DU in different domains on SGD. (Note that only

SGD has unseen schemas in test set.) We observe that the best performances can be obtained in the domains with all seen schemas. The domains that have more partially seen schemas achieve higher accuracies, such as "Hotels", "Movies", "Services". The accuracies decline in the domains with more unseen schemas, such as "Messaging" and "RentalCars". We conclude that Seq2Seq-DU can perform zero-shot learning across domains. However, the ability still needs enhancement.

Domain	Joint GA	Int Acc	Domain	Joint GA	Int Acc
*Messaging**	0.0489	0.3510	*Media**	0.2307	0.9065
*RentalCars**	0.0625	0.7901	*Events**	0.3186	0.9327
*Payment**	0.0719	0.5835	*Hotels***	0.3396	0.9891
*Music**	0.1234	0.9438	*Movies***	0.4386	0.7836
*Restaurants**	0.1295	0.9627	*Travel*	0.4490	0.9966
*Flights**	0.1589	0.9649	*Services***	0.4774	0.9842
*Trains**	0.1683	0.9257	*Alarm**	0.5567	0.5768
*Buses**	0.1684	0.8805	*Weather*	0.5792	0.9965
Homes	0.2275	0.9081	*RideSharing*	0.6702	0.9991

Table 3.8: Accuracy of Seq2Seq-DU in each domain on SGD test set. Domains marked with * are those for which the schemas in the test set are not present in the training set. Domains marked with ** have both the unseen and seen schemas. For other domains, the schemas in the test set are also seen in the training set.

Categorical Slots and Non-categorical Slots

Table 3.9 shows the accuracies of Seq2Seq-DU and the baselines with respect to categorical and non-categorical slots on SGD and MultiWOZ2.2. (We did not compare with FastSGT on SGD dataset due to unavailability of the codes.) One can see that Seq2Seq-DU can effectively deal with both categorical and non-categorical slots. Furthermore, Seq2Seq-DU demonstrates higher accuracies on categorical slots than non-categorical slots. We conjecture that it is due to the co-occurrences of categorical slot values in both the dialogue history and the schema descriptions. The utterance-schema attention can more easily capture the relations between the values.

Model	SGD		MultiWOZ2.2	
	Categorical-Joint-GA	Noncategorical-Joint-GA	Categorical-Joint-GA	Noncategorical-Joint-GA
TRADE	-	-	0.628	0.666
SGD-baseline	0.513	0.361	0.570	0.661
DS-DST	-	-	0.706	0.701
FastSGT	not available	not available	-	-
TripPy	-	-	0.684	**0.733**
Seq2Seq-DU	**0.578**	**0.393**	**0.758**	0.711

Table 3.9: Accuracies of Seq2Seq-DU and baselines with respect to categorical and non-categorical slots on SGD and MultiWOZ2.2.

3.5 Summary

We have proposed a new approach to dialogue state tracking. The approach, referred to as Seq2Seq-DU, takes dialogue state tracking (DST) as a problem of transforming all the utterances in a dialogue into semantic frames (state representations) on the basis of schema descriptions. Seq2Seq-DU is unique in that within the sequence to sequence framework it employs BERT in encoding of utterances and schema descriptions respectively and generates pointers in decoding of dialogue state. Seq2Seq-DU is a global, reprentable, and scalable model for DST as well as NLU (natural language understanding). Experimental results show that Seq2Seq-DU significantly outperforms the state-of-the-arts methods in DST on the benchmark datasets of SGD, MultiWOZ2.2, MultiWOZ2.1, WOZ2.0, DSTC2, M2M, and performs as well as the state-of-the-arts in NLU on the benchmark datasets of ATIS and SNIPS.

Chapter 4

Relation-Guided Natural Language Understanding for Task-Oriented Dialogues

One main problem for natural language understanding in task-oriented dialogues is Dialogue State Tracking (DST). It aims to keep track of users' intentions during the course of a conversation. In DST, modelling the relations among domains and slots is still an under-studied problem. Existing approaches that have considered such relations generally fall short in: (1) fusing prior slot-domain membership relations and dialogue-aware dynamic slot relations explicitly, and (2) generalizing to unseen domains. To address these issues, we propose a novel **D**ynamic **S**chema **G**raph **F**usion **Net**work (**DSGFNet**), which generates a dynamic schema graph to explicitly fuse the prior slot-domain membership relations and dialogue-aware dynamic slot relations. It also uses the schemata to facilitate knowledge transfer to new domains. DSGFNet consists of a dialogue utterance encoder, a schema graph encoder, a dialogue-aware schema graph evolving network, and a schema graph enhanced dialogue state decoder. Empirical results on benchmark datasets (i.e., SGD, MultiWOZ2.1, and MultiWOZ2.2), show that DSGFNet outperforms existing methods.

4.1 Introduction

Task-oriented dialogue systems can help users accomplish different tasks [1], such as flight reservation, food ordering, and appointment scheduling. Conventionally, task-oriented dialogue systems consist of four modules [179]: natural language understanding (NLU), dialogue state tracking (DST), dialogue manager (DM), and natural language generation (NLG). In this thesis, we will focus on the DST module. The goal of DST is to extract users' goals or intentions as dialogue states and keep these states updated over the whole dialogue. In order to track users' goals, we need to have a predefined domain knowledge referred to as a schema, which consists of slot names and their descriptions. Figure 4.1 gives an example of DST in a sample dialogue.

Many models have been developed for DST due to its importance in task-oriented dialogue systems. Traditional approaches use deep neural networks or pre-trained language models to encode the dialogue context and infer slot values from it [38, 39, 4, 36, 40, 41, 42, 40, 45]. These models predict slot values

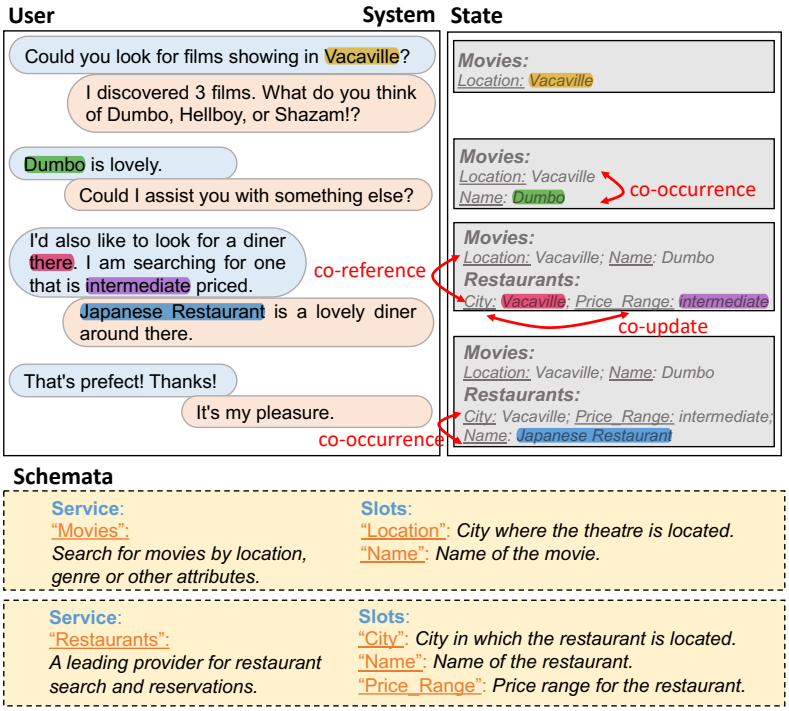


Figure 4.1: An example of DST. Given the schemata for all domains, the slot values are extracted from the user and system utterances (e.g., spans highlighted with the same color in the figure). The dialogue state of each turn is represented as a set of slot-value pairs. Among the domains and slots, there are prior slot-domain membership relations which are expressed in the predefined schemata, and also dialogue-aware dynamic slot relations which depend on the dialogue context (e.g., co-reference, co-update, and co-occurrence).

without considering the relations among domains and slots. However, domains and slots in a dialogue are unlikely to be entirely independent, and ignoring the relations among domains and slots may lead to sub-optimal performance. To address this issue, several recent works have been proposed to model the relations among domains and slots in DST. Some of them introduce predefined schema graphs to incorporate prior slot-domain membership relations, which are defined based on human experience in advance [48, 49]. The others use an attention mechanism to capture dialogue-aware dynamic slot relations [52, 53]. The dialogue-aware dynamic relations are the logical relations of slots across domains, which are highly related to specific dialogue contexts.

However, existing DST models that involve the relations among domains and slots suffer from two major issues: (1) They fail to fuse the prior slot-domain membership relations and dialogue-aware dynamic slot relations explicitly; and (2) They fail to consider their generalizability to new domains. In practical scenarios, task-oriented dialogue systems need to support a large and constantly increasing number of new domains.

To tackle these issues, we propose a novel approach named DSGFNet (Dynamic Schema Graph Fusion Network). For the first issue, DSGFNet dynamically updates the schema graph consisting of the predefined slot-domain membership relations with the dialogue-aware dynamic slot relations. To incorporate the dialogue-aware dynamic slot relations explicitly, DSGFNet adds three new edge types to the schema graph: *co-reference relations*, *co-update relations*, and *co-occurrence relations*. For the second issue, to improve

its generalizability, DSGFNet employs a unified model containing schema-agnostic parameters to make predictions.

Specifically, our proposed DSGFNet comprises of four components: a *BERT-based dialogue utterance encoder* to contextualize the current turn dialogue context and history, a *BERT-based schema graph encoder* to generalize to unseen domains and model the prior slot-domain membership relations on the schema graph, a *dialogue-aware schema graph evolving network* to augment the dialogue-aware dynamic slot relations on the schema graph, and a *schema graph enhanced dialogue state decoder* to extract value spans from the candidate elements considering the evolved schema graph.

The contributions of this chapter can be summarized as follows:

- We improve DST by proposing a dynamic, explainable, and general schema graph which explicitly models the relations among domains and slots based on both prior knowledge and the dialogue context, no matter whether the domains and slots are seen or not.
- We develop a fusion network, DSGFNet, which effectively enhances DST generating a schema graph out of the combination of prior slot-domain membership relations and dialogue-aware dynamic slot relations.
- We conduct extensive experiments on three benchmark datasets (i.e., SGD, MultiWOZ2.1, and MultiWOZ2.2) to demonstrate the superiority of DSGFNet[1] and the importance of the relations among domains and slots in DST.

4.2 Relations among Domains and Slots

The aim of DST is to discern users objectives or intentions as dialogue states and maintain these states updated throughout the conversation. To effectively monitor users goals, a predefined domain knowledge, known as a schema, is essential, comprising slot names and their descriptions. In Figure 3.1, an illustration of DST in a hypothetical dialogue is provided.

There exist two primary types of relations within the predefined domain knowledge. Firstly, there are the prior slot-domain membership relations. For instance, "location" and "name" are examples of prior slot-domain membership for "movie". Secondly, there are the dialogue-aware dynamic slot relations, which can be classified into three types:

1. Co-reference relations: denoting slot pairs with identical values.
2. Co-update relations: indicating slot pairs whose values are updated simultaneously.
3. Co-occurrence relations: representing slot pairs frequently occurring together in the user's requirements.

The rationale behind modeling these relations in dialogue state tracking (DST) lies in enhancing the system's understanding of the ongoing conversation and the user's goals. Here is a breakdown of the intuitions behind each type of relation:

[1]The code is available at `https://github.com/sweetalyssum/DSGFNet`.

Prior slot-domain membership relations: Modeling prior slot-domain membership relations enhances the system's ability to understand the semantic context of the dialogue, improve domain specificity, facilitate slot filling, and support domain adaptation, leading to more accurate and robust dialogue state tracking.

Co-reference relations: By identifying slot pairs with the same value, the system can infer connections between different parts of the dialogue. This helps in maintaining consistency and coherence in the dialogue state representation, as it ensures that relevant information is appropriately linked and considered together.

Co-update relations: When certain slot pairs are consistently updated together, it suggests a strong correlation between them. Modeling these relations allows the system to predict updates more accurately and efficiently, reducing ambiguity and improving the overall robustness of the dialogue state.

Co-occurrence relations: Slot pairs that frequently appear together in user needs often indicate dependencies or associations between them. By recognizing these co-occurring slots, the system can anticipate user preferences or requirements more effectively, enabling more personalized and contextually relevant responses.

Overall, modeling these relations in DST enables the system to capture the nuanced dynamics of the conversation, leading to more accurate understanding and effective response generation.

The labels of these relations in dialogue state tracking (DST) are typically obtained through various methods, including manual annotation and rule-based approaches. Here is how each relation type can be labeled:

(1) Prior slot-domain membership relations: These relations are predefined by the domain expertise. (2) Co-reference relations: These relations involve identifying slot pairs with the same value. The labeling process often involves comparing slot values across turns in the dialogue. If two slot values match or are sufficiently similar, they are labeled as co-referent. (3) Co-update relations: Labeling co-update relations involves observing how slot values change over the course of the dialogue. If certain slot pairs are consistently updated together in response to user input or system actions, they are labeled as co-updating. This can be identified through manual annotation or by analyzing patterns in the dialogue data. (4) Co-occurrence relations: Identifying co-occurrence relations entails recognizing slot pairs that frequently appear together in the user's needs or preferences. This can be determined through statistical analysis of dialogue corpora, looking for patterns where certain slot pairs tend to occur together in user utterances related to a specific domain.

In this chapter, we proposed a Dynamic Schema Graph Fusion Network to learn and model these relations. More details are in the following sections.

4.3 Dynamic Schema Graph Fusion Network

The proposed DSGFNet consists of four components: (1) a *BERT-based dialogue utterance encoder* that aims to contextualize the tokens of the current turn and the dialogue history; (2) a *schema graph encoder* that is able to generalize to unseen domains and shares information among predefined slot-domain membership relations; (3) a *dialogue-aware schema graph evolving*

network that adds the dialogue-aware dynamic slot relations into the schema graph; and (4) a *schema graph enhanced dialogue state decoder* that extracts the value span from the candidate elements based on the evolved schema graph. Figure 4.2 illustrates the architecture.

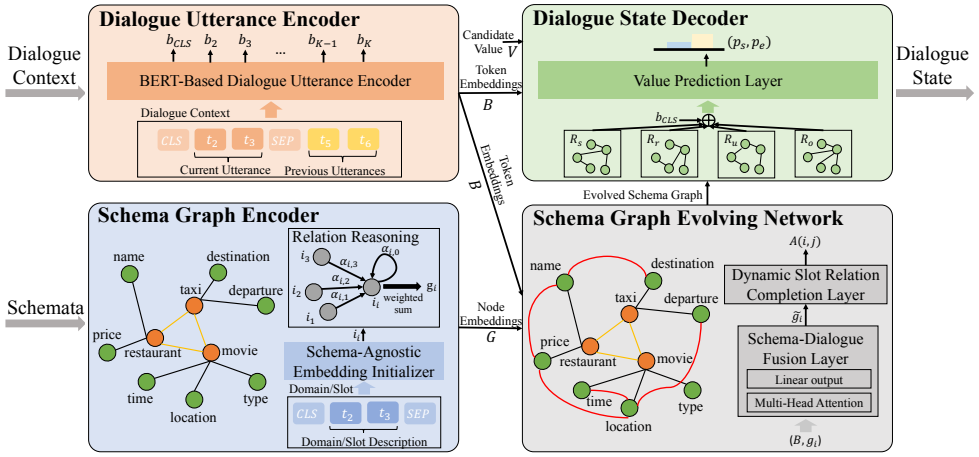


Figure 4.2: The architecture of DSGFNet, which contains a dialogue utterance encoder, a schema graph encoder, a schema graph evolving network, and a dialogue state decoder.

4.3.1 Dialogue Utterance Encoder

This encoder takes as input the current and previous dialogue utterances. Specifically, the input is a sequence of tokens with length K, i.e., $[t_1, ..., t_K]$. Here, we set the first token t_1 to `[CLS]`; subsequent are the tokens in the current dialogue utterance and the ones in the previous dialogue utterances, which are separated by `[SEP]`. We employ BERT [43] to obtain contextual token embeddings. The output is a tensor of all the token embeddings $\boldsymbol{B} = [\boldsymbol{b}_1, ..., \boldsymbol{b}_K]$, with one embedding for each token.

4.3.2 Schema Graph Encoder

To make use of the slot-domain membership relations defined by prior domain knowledge, we construct a schema graph based on the predefined ontology. In task-oriented dialogue systems, the predefined ontology is used to represent the user's needs for the specific task. An example is shown in Figure 4.2. In this schema graph, each node represents either a domain or a slot, and all the slot nodes are connected to their corresponding domain nodes. In order to allow information propagation across domains, all the domain nodes are connected with each other.

Schema-Agnostic Embedding Initializer. To generalize to unseen domains, DSGFNet initializes the schema graph node embeddings via a schema-agnostic projection. Inspired by zero-shot learning [180], we propose a schema-agnostic embedding initializer to project schemata across domains into a unified semantic distribution. Even though the domain is unseen, the model can still understand the domain using the unified semantic distribution. Specifically, we feed a natural language description of one slot/domain into BERT, using the output of `[CLS]` as the semantic embeddings for this slot/domain. The semantic embeddings for the set of slot and domain is $\boldsymbol{I} = [\boldsymbol{i}_1, ..., \boldsymbol{i}_{N+M}]$, where

N and M are the number of slots and domains, respectively. We constrain the schema embedding initializer not to have any domain-specific parameters so that it can generalize to unseen domains.

Slot-Domain Membership Relation Reasoning Network. To involve the prior slot-domain membership relations into the schema graph node embeddings, DSGFNet propagates information among slots and domains over the schema graph. We add a self-loop to each node because the nodes need to propagate information to themselves. Inspired by the GAT model [181], we propose a slot-domain membership relation reasoning network to propagate information over the schema graph. For each node, we first compute attention scores $\boldsymbol{\alpha}$ for its neighbours. These attention scores are used to weigh the importance of each neighboring node. Formally, the attention scores are calculated as follows:

$$h_{i,j} = \text{ReLU}(\mathbf{W}^\top \cdot [\boldsymbol{i}_i, \boldsymbol{i}_j]), \alpha_{i,j} = \frac{\exp(h_{i,j})}{\sum_{k \in \mathcal{N}_i} \exp(h_{i,k})}, \tag{4.3.1}$$

where $\mathbf{W}$ is a matrix of parameters and $\mathcal{N}_i$ is the neighborhood of the i-th node. The normalized attention coefficients and the activation function are used to compute a non-linear weighted combination of the neighbours. This is used to compute the tensor of the schema graph node embeddings $\boldsymbol{G} = (\boldsymbol{g}_1, ..., \boldsymbol{g}_{N+M})$:

$$\boldsymbol{g}_i = \text{ReLU}\left(\sum_{j \in \mathcal{N}_i} \alpha_{i,j} \cdot \boldsymbol{i}_j\right), \tag{4.3.2}$$

where $i \in \{1, \ldots, N+M\}$. To explore the higher-order connectivity information of slots across domains, we stack l layers of the reasoning network. Each layer takes the node embeddings from the previous layer as input, and outputs the updated node embeddings to the next layer.

4.3.3 Schema Graph Evolving Network

We propose a schema graph evolving network to incorporate the dialogue-aware dynamic slot relations into the schema graph, which is composed of two layers, a schema-dialogue fusion layer and a dynamic slot relation completion layer.

Schema-Dialogue Fusion Layer. Since the dynamic slot relations are related to the dialogue context, we need to fuse the dialogue context information into the schema graph. We adopt the multi-head attention [182] to achieve this goal. The mathematical formulation is:

$$\boldsymbol{H} = \text{MultiHead}(\text{Q} = \boldsymbol{g}_i, \text{K} = \boldsymbol{B}, \text{V} = \boldsymbol{B}), \tag{4.3.3}$$

$$\tilde{\boldsymbol{g}}_i = \boldsymbol{H} \cdot \boldsymbol{W}_a, \tag{4.3.4}$$

where $\boldsymbol{W}_a$ is learnable parameters of a linear projection after the multi-head attention, and $\tilde{\boldsymbol{g}}_i$ is the dialogue-aware schema graph node embeddings.

Dynamic Slot Relation Completion Layer. This layer aims to augment the dynamic slot relations on the schema graph based on the dialogue-aware node embeddings. To involve the dialogue-aware dynamic slot relations into DST explicitly, DSGFNet defines three types of dynamic slot relations: (1)

Co-reference relations occur when a slot value has been mentioned earlier in the dialogue and has been assigned to another slot; (2) Co-update relations occur when slot values are updated together at the same dialogue turn, and; (3) Co-occurrence relations occur when slots with a high co-occurrence probability in a large dialogue corpus appear together in the current dialogue. Specifically, we feed the dialogue-aware slot node representations into a multi-layer perceptron followed by a 4-way softmax function to identify the relations between slot pairs, which include the $none$ relation and the three dynamic relations mentioned above. Formally, given the i-th and j-th dialogue-aware slot node embeddings $\tilde{\boldsymbol{g}}_i$ and $\tilde{\boldsymbol{g}}_j$, we obtain an adjacent matrix of the dynamic slot relations for all slot pairs as follows:

$$\boldsymbol{A}(i,j) = \arg\max\left(\text{softmax}(\text{MLP}(\tilde{\boldsymbol{g}}_i \oplus \tilde{\boldsymbol{g}}_j))\right), \tag{4.3.5}$$

where $\oplus$ is the concatenation. With $\boldsymbol{A}$, we add dynamic slot relation edges to the schema graph.

4.3.4 Dialogue State Decoder

To decode the slot values by means of incorporating the slot-domain membership relations and dialogue-aware dynamic slot relations which are captured by the evolved schema graph, we propose a schema graph enhanced dialogue state decoder.

To learn a more comprehensive slot node embedding, we need to fuse multiple relations on the evolved schema graph. DSGFNet divides different relations on the schema graph into sub-graphs R_s, R_r, R_u, R_o, which represent slot-domain membership relation, co-reference relation, co-update relation, and co-occurrence relation, respectively. For each sub-graph R_i, its node embeddings $\boldsymbol{s}_i$ are obtained by attending over the neighbors, which is the same as the method used in Section 4.3.2. Considering that different relation types have different contributions to the node interactions for different dialogue contexts [183], we aggregate these different sub-graphs via an attention mechanism as follows:

$$\boldsymbol{S} = [\boldsymbol{s}_s; \boldsymbol{s}_r; \boldsymbol{s}_u; \boldsymbol{s}_o], \tag{4.3.6}$$

$$\boldsymbol{\beta} = \text{softmax}(\boldsymbol{S}^\top \cdot \tanh(\boldsymbol{W}_s \cdot \boldsymbol{b}_{[CLS]} + \boldsymbol{b}_s)), \tag{4.3.7}$$

$$\boldsymbol{s} = \boldsymbol{S} \cdot \boldsymbol{\beta}, \tag{4.3.8}$$

where $\boldsymbol{W}_s$, $\boldsymbol{b}_s$ are learnable weights, $\boldsymbol{b}_{[CLS]}$ is the output of BERT-based dialogue utterance encoder.

Each slot value is extracted by a value predictor based on the corresponding fused slot node embeddings $\boldsymbol{s}$. The value predictor is a trainable nonlinear classifier followed by two parallel softmax layers to predict start and end positions in candidate elements $\boldsymbol{C}$, which are composed by the dialogue context $\boldsymbol{B}$ and slots' candidate value vocabulary $\boldsymbol{V}$:

$$\boldsymbol{C} = [\boldsymbol{B}; \boldsymbol{V}] \tag{4.3.9}$$

$$[\boldsymbol{l}_s, \boldsymbol{l}_e] = \boldsymbol{r}_d \cdot \tanh(\boldsymbol{s}^\top \cdot \boldsymbol{W}_d \cdot \boldsymbol{C} + \boldsymbol{b}_d), \tag{4.3.10}$$

$$p_s = \text{softmax}(\boldsymbol{l}_s), \tag{4.3.11}$$

$$p_e = \text{softmax}(\boldsymbol{l}_e), \tag{4.3.12}$$

where $\boldsymbol{r}_d$, $\boldsymbol{W}_d$, and $\boldsymbol{b}_d$ are trainable parameters. Note that if the end position is before the start position, the resulting span will simply be "None". If the start position is in the slots candidate value vocabulary, the resulting span will only pick the candidate value in this position.

4.3.5 Training and Inference

Cross-entropy between predicted value span $[p_s, p_e]$ and ground truth value span is utilized to measure the loss of the value span prediction $\mathcal{L}_s$. The dynamic slot relation identifier is optimized by the cross-entropy loss $\mathcal{L}_r$ between predicted dynamic relation $\boldsymbol{A}$ and the ground truth dynamic slot relation. We train dialogue state decoder and dynamic slot relation identifier together, the joint loss $\mathcal{L}$ is computed as follows:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_r + (1 - \lambda) \cdot \mathcal{L}_s, \tag{4.3.13}$$

where $\lambda \in [0, 1]$ is a balance coefficient. During inference, the predicted dynamic slot relation $\boldsymbol{A}$ is used to predict value span as dialogue state.

4.4 Experiments

4.4.1 Datasets

We conduct experiments on three task-oriented dialogue benchmark datasets: SGD [55], MultiWOZ2.2 [164], and MultiWOZ2.1 [184]. Among them, SGD is by far the most challenging dataset which contains over 16,000 conversations between a human-user and a virtual assistant across 16 domains. Unlike the other two datasets, it also includes unseen domains in the test set. MultiWOZ2.2 and MultiWOZ2.1 are smaller human-human conversations benchmark datasets, which contain over 8,000 multi-turn dialogues across 8 and 7 domains, respectively. MultiWOZ2.2 is a revised version of MultiWOZ2.1, which is re-annotated with a different set of annotators and also canonicalized entity names. Details of datasets are provided in Table 4.1.

Table 4.1: Characteristics of the datasets in experiments. The numbers provided are for the training sets of the corresponding datasets.

Characteristics	**SGD**	**MultiWOZ2.2**	**MultiWOZ2.1**
No. of domains	16	8	7
No. of dialogues	16,142	8,438	8,438
Total no. of turns	329,964	113,556	113,556
Avg. turns per dialogue	20.44	13.46	13.46
Avg. tokens per turn	9.75	13.13	13.38
No. of slots	215	61	37
Unseen domains in test set	Yes	No	No

4.4.2 Baselines

We compare with the following existing models, which are divided into two categories.

(1) Models that can predict dialogue state on unseen domains: *SGD-baseline* [55], a schema-guided paradigm that predicts states for unseen domains; *FastSGT* [56], a BERT-based model that uses multi-head attention projections to analyze dialogue; *Seq2Seq-DU* [52], a sequence-to-sequence framework which decodes dialogue states in a flatten format. The SGD-baseline is implemented using its open-source code. For FastSGT, we use the results reported by the authors. The Seq2seq-DU method is implemented by us.

(2) Models that cannot predict dialogue state on unseen domains: *TRADE* [4], a generation model which generates dialogue states from utterances using a copy mechanism; *DS-DST* [40], a dual strategy that classifies over a picklist or finding values from a slot span; *TripPy* [53], an open-vocabulary model which copies values from dialogue context, or slot values in previous dialogue state; *SOM-DST* [185], a selectively overwriting mechanism which first predicts state operation on each of the slots and then overwrites with new values; *MinTL-BART* [46], a plug-and-play pre-trained model which jointly learns dialogue state tracking and dialogue response generation; *SST* [48], a graph model which fuses information from utterances and static schema graph; *PPTOD* [186], a multi-task pre-training strategy that allows the model to learn the primary TOD task completion skills from heterogeneous dialog corpora. We reproduce the baselines using their open-source code.

4.4.3 Evaluation Measures

Our evaluation metrics are consistent with prior works on these datasets [55, 56, 4, 40]. We compute the Joint Goal Accuracy (Joint GA) on all test sets for straightforward comparison with the state-of-the-art methods. Joint GA is defined as the ratio of dialogue turns for which all slots have been filled with the correct values according to the ground truth.

A paired t-test is conducted to determine whether there is a significant difference between our method and the best baseline. On the SGD and MultiWOZ2.2 dataset, we compare our method with Seq2Seq-DU. For MultiWOZ2.1, we compare our method with PPTOD. The significance level (p-value) is set at 0.05.

4.4.4 Training

We use BERT (i.e., BERT-base and uncased) to encode utterances and schema descriptions. The BERT models are fine-tuned in the training process. The maximum length of an input sequence is set to 512. The hidden size of the schema graph encoder and the schema graph evolving network is set to 256. The dropout probability is 0.3. The balance coefficient λ is 0.5. Adam [178] is used for optimization with an initial learning rate (LR) of 2e-5. We conduct training with a warm-up proportion of 10% and let the LR decay linearly after the warm-up phase.

4.5 Results and Discussion

Tables 4.2, 4.3, 4.4 show the performance of DSGFNet as well as the baselines on the three datasets respectively. It is shown that DSGFNet achieves state-

of-the-art performance in unseen domains on SGD, all domains on SGD, and MultiWOZ2.2. All improvements observed compared to the baselines are statistically significant according to two sided paired t-test ($p < 0.05$). And the performance on MultiWOZ2.1 are comparable with the state-of-the-art[2]. Most notably, DSGFNet improves the performance on SGD most significantly, which has unseen domains and more complex schemata domains, compared to the runner-up. It indicates that DSGFNet can facilitate knowledge transfer to new domains and improve relation construction among complex schemata domains. We conjecture that it is due to DSGFNet containing the schema-agnostic encoder and dynamic schema graph. The following analysis provides a better understanding of our models strengths.

Table 4.2: Joint GA of DSGFNet and baselines in unseen domains and all domains on SGD dataset. DSGFNet significantly improves over the best baseline Seq2Seq-DU (two-sided paired t-test, $p < 0.05$).

Models	**SGD Unseen Domains**	**SGD All Domains**
SGD-baseline [55]	20.0%	25.4%
FastSGT [56]	20.8%	29.2%
Seq2Seq-DU [52]	23.5%	30.1%
DSGFNet	**24.4%**	**32.1%**

Table 4.3: Joint GA of DSGFNet and baselines on MultiWOZ2.2. DSGFNet significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$).

Model	**MultiWOZ2.2**
SGD-baseline [55]	42.0%
TRADE [4]	45.4%
DS-DST [40]	51.7%
TripPy [53]	53.5%
Seq2Seq-DU [52]	54.4%
DSGFNet	**55.8%**

4.5.1 Ablation Study

We conduct an ablation study on DSGFNet to quantify the contributions of various factors: the usage of slot-domain membership relations, dynamic slot relations, and multiple relation aggregation. The results indicate that the dynamic schema graph of DSGFNet is indispensable for DST.

Effect of Slot-Domain Membership Relations

To check the effectiveness of the slot-domain membership relations, we remove the schema graph by replacing the prior slot-domain relation adjacency matrix with an identity matrix $\boldsymbol{I}$. Results in Table 4.5 show that the joint goal

[2]TRADE, SST use the original MultiWOZ datasets. The other models use the data preprocessed by TripPy.

Table 4.4: Joint GA of DSGFNet and baselines on MultiWOZ2.1. DSGFNet achieves comparable performance of the best baseline.

Model	MultiWOZ2.1
SGD-baseline [55]	43.4%
TRADE [4]	46.0%
DS-DST [40]	51.2%
SOM-DST [185]	53.0%
MinTL-BART [46]	53.6%
SST [48]	55.2%
TripPy [53]	55.3%
PPTOD [186]	**57.1%**
DSGFNet	56.7%

accuracy of DSGFNet without the slot-domain membership relations decreases markedly on unseen domains of SGD, all domains of SGD, MultiWOZ2.2, and MultiWOZ2.1. It indicates the schema graph, which contains slot-domain membership relations, can facilitate knowledge sharing among domain and slot no matter whether the domain is seen or not.

Effect of Dynamic Slot Relations

To investigate the effectiveness of the dialogue-aware dynamic slot relations in the schema graph, we eliminate the evolving network of DSGFNet. Table 4.5 shows the results on unseen domains of SGD, all domains of SGD, MultiWOZ2.2, and MultiWOZ2.1 in terms of joint goal accuracy. One can observe that without the dynamic slot relations the performance deteriorates considerably. In addition, there is a more markedly performance degradation compared with the results of the slot-domain membership relations. It indicates that the dynamic slot relations are more essential for DST, which can facilitate the understanding of the dialogue context.

Effect of Multiple Relation Aggregation

To validate the effectiveness of the schema graph relation aggregation mechanism in the dialogue state decoder, we directly concatenate all sub-graph representations instead of calculating a weighted sum via the sub-graph attention. As shown in Table 4.5, the performance of the models without the relation aggregation layer in terms of joint goal accuracy decreases markedly compared to DSGFNet. It indicates that the attentions to different types of relations affect the dialogue understanding ability.

Table 4.5: Ablation study on unseen domains of SGD, all domains of SGD, MultiWOZ2.2 and MultiWOZ2.1.

Model	Joint GA Unseen Domains SGD	Joint GA All Domains SGD	Joint GA MultiWOZ 2.2	Joint GA MultiWOZ 2.1
DSGFNet	24.4%	32.1%	55.8%	56.7%
-w/o Slot-Domain Membership Relations	21.9%	29.8%	53.4%	54.1%
-w/o Dynamic Slot Relations	20.6%	28.6%	52.2%	53.2%
-w/o Relation Aggregation	23.8%	31.5%	55.2%	55.9%

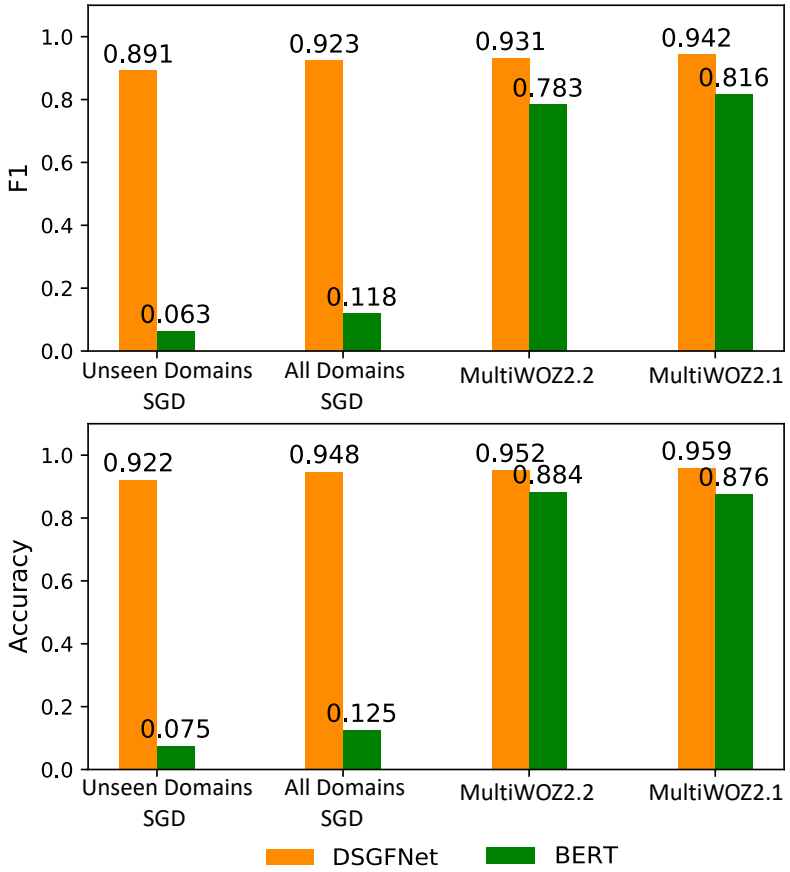


Figure 4.3: F1 and Accuracy of DSGFNet and BERT for dynamic relation prediction on unseen domains SGD, all domains of SGD, MultiWOZ2.2 and MultiWOZ2.1.

4.5.2 Further Analysis

Prediction of Dynamic Slot Relations

In order to test the discriminative capability of DSGFNet for dynamic slot relations, we evaluate the performance of the schema graph evolving network. Since baselines cannot predict the dynamic slot relations explicitly, we compare DSGFNet with the BERT-based classification approach. Following the classification task in BERT, the input sequence starts with [CLS], followed by the tokens of the dialogue context and slot pairs, separated by [SEP], and the [CLS] representation is fed into an output layer for classification. Figure 4.3 shows the results on unseen domains of SGD, all domains of SGD, MultiWOZ2.2, and MultiWOZ2.1 in terms of F1 and Accuracy for slot relations. From the results, we observe that DSGFNet outperforms BERT significantly. We conjecture that it is due to the exploitation of schema graph with slot-domain membership relations in DSGFNet. In addition, since BERT without schema encoder cannot solve unseen domains, there is a significant performance degradation on SGD which contains a large number of unseen domains in the test set.

Effects of Each Type of Dynamic Slot Relation

To better illustrate the effectiveness of augmenting slot relations on the schema graph, we study how different dynamic slot relations affect the DST perfor-

Table 4.6: Case study of DSGFNet and Seq2Seq-DU on SGD. Slot values are extracted from the dialogue context with the same color. The relation of yellow high light slot pair is predicted as co-reference. The relation of red underline slot pair is predicted as co-update. The relation of bold font slot pair is predicted as co-occurrence. Slot values in red high light are incorrectly predicted ones.

Dialogue Utterance	[*User*]: What's the weather going to be like in vancouver on March 10th? [*Sys*]: The forecast average is 68 degrees with a 25 per cent chance of rain. [*User*]: Any good attractions in town? [*Sys*]: I have 10 good options including Bloedel Conservatory, a city park. [*User*]: Lovely! Can you book me a ride there? [*Sys*]: Do you want a luxury or pool ride? How many people? [*User*]: Just a regular ride please, book for 1. [*Sys*]: Confirming you want to book a regular cab to Bloedel Conservatory for 1 person.
Ground Truth Dialogue State	[*Weather*]: city = "vancouver"; date = "March 10th"; [*Travel*]: location = "vancouver"; [*RideSharing*]: destination = "Bloedel Conservatory"; number of seats = "1"; ride type = "regular";
State Predictions of DSGFNet	[*Weather*]: **city** = "vancouver"; **date** = "March 10th"; [*Travel*]: location = "vancouver"; [*RideSharing*]: destination = "Bloedel Conservatory"; number of seats = "1"; ride type = "regular";
State Predictions of Seq2seq-DU	[*Weather*]: city = "vancouver"; date = "March 10th"; [*Travel*]: location= "town"; [*RideSharing*]: destination = "Bloedel Conservatory"; number of seats = "1"; ride type = none;

Table 4.7: Performance comparison with different dynamic slot relations and fully-connected relations on unseen domains of SGD, all domains of SGD, MultiWOZ2.2 and MultiWOZ2.1.

Model	Joint GA Unseen Domains SGD	Joint GA All Domains SGD	Joint GA MultiWOZ 2.2	Joint GA MultiWOZ 2.1
-w All Dynamic Relations	24.4%	32.1%	55.8%	56.7%
-w Co-reference Relation	21.5%	29.8%	53.9%	54.7%
-w Co-occurrence Relation	23.8%	31.7%	55.3%	55.9%
-w Co-update Relation	22.3%	30.1%	53.5%	54.5%
-w/o Dynamic Relations	20.6%	28.6%	52.2%	53.2%
-w Fully-connected Relations	21.3%	29.9%	54.2%	54.9%

mance. Table 4.7 presents the joint goal accuracy of DSGFNet with different dynamic relations on unseen domains of SGD, all domains of SGD, MultiWOZ2.2, and MultiWOZ2.1. One can see that the performance of DSGFNet with each type of dynamic slot relation surpasses that without any dynamic slot relations considerably. Thus, all types of dynamic slot relations in the schema graph are helpful for dialogue understanding. Furthermore, the performance of DSGFNet with co-occurrence relation is superior to the performance with the other two dynamic slot relations. We conjecture that it is due to the fact that a large percentage of dynamic relations is the co-occurrence relation, which has an incredible effect on DST.

Effect of Automatic Relation Completion

To demonstrate the effectiveness of automatically completing each type of slot relations on the schema graphs, we replace four automatically-completed subgraphs in DSGFNet with four fully-connected graphs. As shown in Table 4.7, the performance of the model with the fully-connected graphs in terms of joint goal accuracy decreases significantly compared to DSGFNet (two-sided paired t-test, $p < 0.05$). We believe that this is caused by the noise introduced

by the redundancy captured by the relations between all pairs of slots. In addition, sampling the relations using our strategy can also reduce the memory requirements when the number of slots and domains are large.

Case Study

We make qualitative analysis on the results of DSGFNet and Seq2seq-DU on SGD. We find that DSGFNet can make a more accurate inference of dialogue states by using the dynamic schema graph. For example, as shown in Table 4.6, "city-location" is predicted as co-reference relation, "city-date" and "number of seats" and "ride-type" are predicted as co-update relation, "city-date" is predicted as co-occurrence relation. Based on the dynamic schema graph, DSGFNet propagates information involving slot-domain membership relations and dynamic slot relations. Thus, it infers slot values more correctly. In contrast, since Seq2seq-DU ignores the dynamic slot relations, it cannot properly infer the values of "location" and "ride-type", which have dynamic slot relations with other slots.

Dynamic Slot Relations Label Collection

Dynamic schema graph in DSGFNet has three types of dynamic slot relations, which includes co-reference relations, co-update relations and co-occurrence relations. The labels of these relations are used to train the schema graph evolving network. We first collected all possible slot pairs from ground truth dialogue state for each dialogue turn. And then we labeled these slot pairs by the rules as follows: (1) If one slot value has been assigned to another slot in earlier turn of the dialogue, we label the relation between these two slots as co-reference. (2) If values of two slot in the same dialogue turn are updated together, we label the relation between these two slots as co-update. (3) If the co-occurrence probability of two slots in the training set of SGD, MultiWOZ2.1, and MultiWOZ2.2 is higher than 5%, we label the relation between these two slots as co-occurrence. Table 4.8 shows the the proportion of different types of dynamic slot relations on these datasets.

Table 4.8: The proportion of different types of dynamic slot relations on SGD, MultiWOZ2.2, and MultiWOZ2.1 in training sets.

Relation	**SGD**	**MultiWOZ2.2**	**MultiWOZ2.1**
Co-reference	5.11%	4.21%	4.29%
Co-update	9.31%	4.01%	4.13%
Co-occurrence	31.13%	37.53%	36.53%

Performance on Different Domains

We further investigate the performance of DSGFNet on different domains. Table 4.9 shows the joint goal accuracy of DSGFNet in different domains on SGD. We observe that the presence of schemata in the training data is the major factor affecting the performance. We see that the best performance can be obtained in the domains with all seen schemata. The domains which have partially unseen schemata achieve higher accuracy, such as "Hotels", "Movies",

Table 4.9: Accuracy of DSGFNet in each domain on SGD test set. Domains marked with * are those for which the schemata in the test set are not present in the training set. Domains marked with ** have both the unseen and seen schemata. For other domains, the schemata in the test set are also seen in the training set.

Domain	Joint GA	Domain	Joint GA
*RentalCars**	5.11%	*Homes*	22.46%
*Messaging**	5.48%	*Events**	32.02%
*Payment**	7.31%	*Hotels***	33.13%
*Music**	11.87%	*Movies***	42.13%
*Buses**	12.72%	*Services***	45.39%
*Trains**	16.39%	*Travel*	48.30%
*Flights**	16.64%	*Alarm**	53.27%
*Restaurants**	17.01%	*RideSharing*	56.42%
*Media**	20.83%	*Weather*	68.49%

and "Services" domains. The accuracy declines in the domains with only unseen schemata, such as "RentalCars" and "Messaging". However, among the domains with only unseen schemata, those have similar schemata to training data resulting in superior performance, such as "Alarm" and "Events" domains. We conclude that DSGFNet is able to perform zero-shot learning and share knowledge across domains. However, more sharing of information should be utilized to enhance the generalization ability.

Analysis of Parameters in DSGFNet

We further investigate the impacts of parameter settings on the performance of DSGFNet on SGD, MultiWOZ2.2, and MultiWOZ2.1. We validate the effects of four factors: the layer of propagation on the schema graph, the number of selected dialogue turns used in the schema-dialogue fusion layer, the layer of MLP in the dynamic slot relation completion layer, and the balance coefficient λ in the loss function. Figures 4.4, 4.5, 4.6, 4.7 show the results of DSGFNet with varying parameters on SGD, MultiWOZ2.2, and MultiWOZ2.1 in terms of joint goal accuracy. We observe that the optimal layer of propagation is not consistent across datasets. It seems that 3 is desired in more datasets. In addition, DSGFNet demonstrates the best performance when leveraging full dialogue history. We conjecture that it is due to that the incomplete dialogue history leads to confusing information. Moreover, 8 layers MLP for relation completion obtains the optimal performance over three datasets. Furthermore, the optimal performance is consistently achieved when the balance coefficient λ is around 0.5.

4.6 Summary

We have proposed a new approach to DST, referred to as DSGFNet, which effectively fuses prior slot-domain membership relations and dialogue-aware dynamic slot relations on the schema graph. To incorporate the dialogue-aware

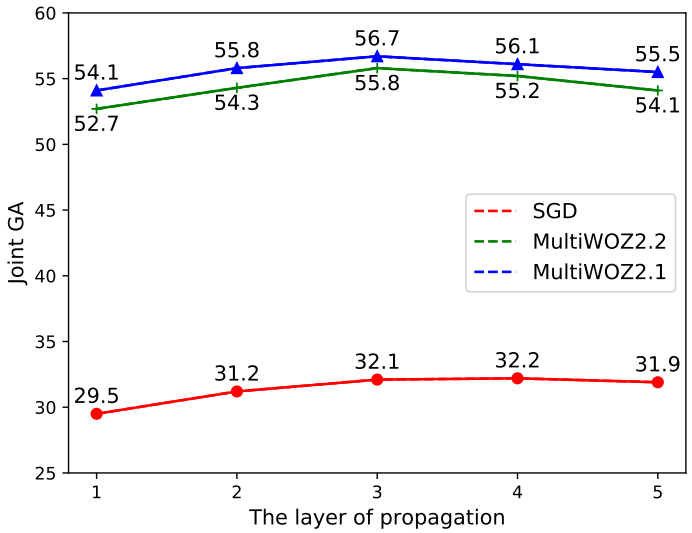


Figure 4.4: Performance comparison *w.r.t.* the layer of propagation on the schema graph.

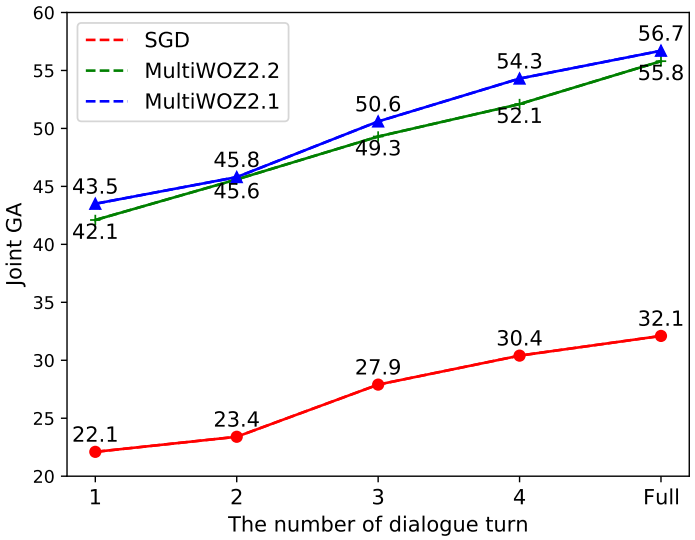


Figure 4.5: Performance comparison *w.r.t.* the number of dialogue turns used in the schema-dialogue fusion layer.

dynamic slot relations into DST explicitly, DSGFNet identifies co-reference, co-update, and co-occurrence relations. To improve the generalization ability, DSGFNet employs a schema-agnostic graph attention network to share information. Experimental results show that DSGFNet outperforms the existing methods in DST on three benchmark datasets, including unseen domains of SGD, all domains of SGD, MultiWOZ2.1, and MultiWOZ2.2. For future work, we intend to further enhance our approach by utilizing more complex schemata and data augmentation techniques.

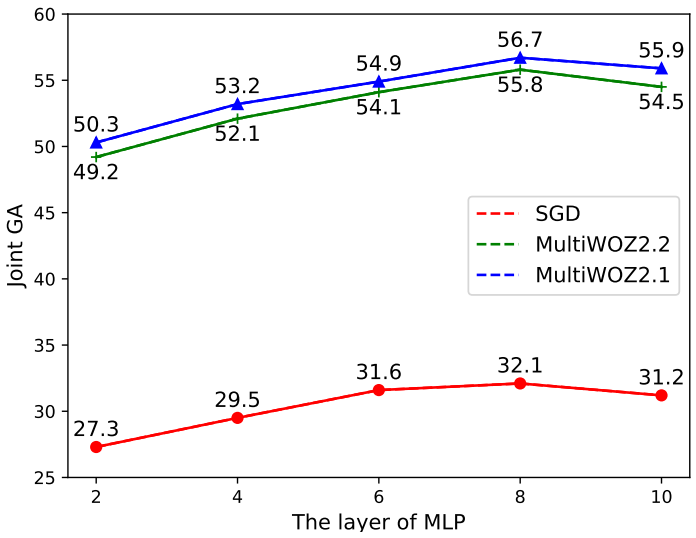


Figure 4.6: Performance comparison *w.r.t.* the layer of MLP in the dynamic slot relation completion layer.

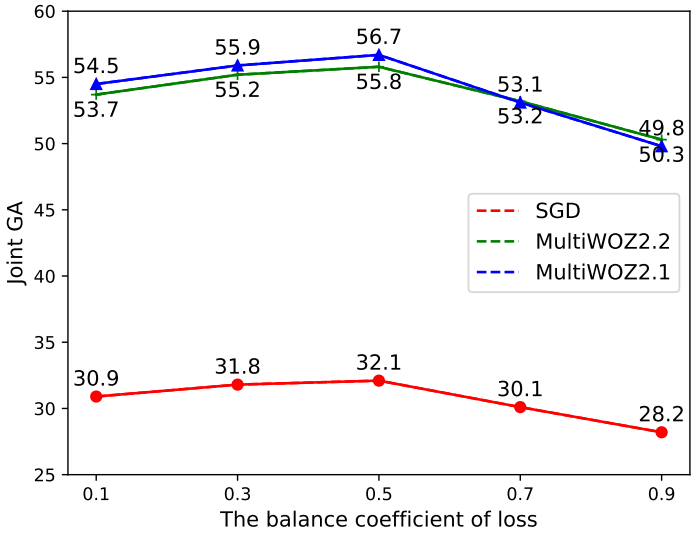


Figure 4.7: Performance comparison *w.r.t.* the balance coefficient in the loss function.

Chapter 5

Document-Guided Natural Language Generation for Task-Oriented Dialogues

Task-oriented dialogue systems aim at providing users with task-specific services. Users of such systems often do not know all the information about the task they are trying to accomplish, requiring them to seek information about the task. To provide accurate and personalized task-oriented information-seeking results, task-oriented dialogue systems need to address two potential issues: 1) users' inability to describe their complex information needs in their requests; and 2) ambiguous/missing information the system has about the users. To address these issues, task-oriented dialogue systems must have the ability to proactively ask questions so as to *clarify* necessary information. In this chapter, we propose a Multi-Attention Large Language Model framework, named MALLM, which can ask questions to clarify the user's information needs and the user's profile for information-seeking on task-oriented dialogues. MALLM consists of a text encoder, an answer confidence embedding network, and a natural language decoder. In particular, the answer confidence embedding network takes into account the task knowledge to derive confidence of the predicted answer to the user's information-seeking request. Our experimental results show that MALLM outperforms baselines on both clarification question generation and answer prediction. To foster research in this area, we have made the code publicly available[1].

5.1 Introduction

The primary goal of a task-oriented dialogue system is to help the user complete a task. Since most tasks can be highly complex, and users often lack background knowledge about the tasks, users have to search for task-related information to accomplish tasks. This type of information seeking behaviour is referred to as task-oriented information seeking and is an important problem that needs to be solved by task-oriented dialogue systems [187, 93, 188].

During the information-seeking process in task-oriented dialogues, users often fail to formulate their complex task-related information needs in a single request. Furthermore, the system may not have enough information about

[1]Code is available at https://github.com/sweetalyssum/CQG_code.

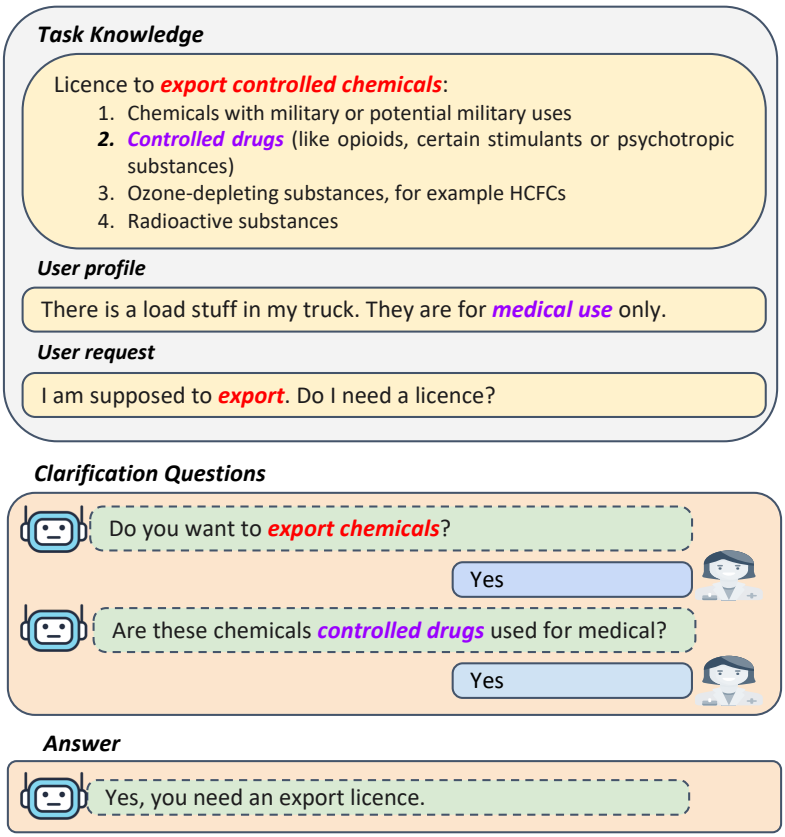


Figure 5.1: An example of task-oriented information-seeking dialogue. A user needs to search for information related to a specific task. Due to the unclear user request and user profile, the task-oriented dialogue system should ask clarification questions to clarify the user request and the user profile based on the task knowledge to provide the answer.

the profile of the user to accurately respond to the user's request. In order to provide accurate and personalized task-related information-seeking results, systems have to ask questions to clarify user request and user profile. Figure 7.1 shows an example of task-oriented information-seeking dialogue.

Many methods have been proposed for asking clarification questions for general information seeking tasks, which include ranking-based models [189, 87, 86] and generation-based models [90, 88, 89]. Ranking based models assume the existence of a candidate set of clarification questions and cast clarification question generation as a ranking problem. Generation based methods tend to be more challenging as they do not assume the existence of candidate questions.

Compared to asking clarification questions for general information seeking, two issues need to be addressed for task-oriented information seeking: 1) user's information needs are related to a specific task. Existing task-oriented systems assume that the system has access to some prior knowledge about the task [190, 98]. Therefore, a task-oriented information-seeking model needs to ask clarification questions considering the task knowledge the system has, and;

2) for different user information, the answers may be different [191]. Task-oriented dialogue systems should provide personalized answers by incorporating user profiles. Therefore, task-oriented information-seeking model needs to ask clarification questions considering the user profile.

In this chapter, we focus on the problem of clarification question generation for task-oriented information seeking. We propose a Multi-Attention Large Language Model framework (MAS2S), which considers task knowledge to clarify the user request and user profile for information seeking on task-oriented dialogues. Specifically, MAS2S consists of a text encoder, a final response confidence embedding network, and a natural language decoder. At each turn of the dialogue, the text encoder transforms the user request, user profile, task knowledge, and dialogue history into semantic embeddings. The answer confidence embeddings network creates knowledge-aware dialogue representations and knowledge-aware user representations using the attention mechanism to calculate answer confidence embeddings. The natural language decoder sequentially generates a clarification question or an answer for the user request on the basis of the answer confidence embeddings.

We conduct experiments on dataset ClariT for asking clarifying questions for information seeking in task-oriented dialogues. We compare MAS2S with competitive clarification question generation models for general purpose information seeking and natural language generation models in task-oriented dialogue systems. Experiment results show that MAS2S significantly outperforms all the baselines. Extensive analyses also reveal the significance of clarifying user requests and user profiles based on task knowledge in task-oriented information seeking.

Our contributions can be summarized as follows:

- We formulate a new problem of asking clarifying questions for information seeking in task-oriented dialogues. Given a task knowledge, a user profile, and a user request, the system should clarify the user request and the user profile to provide a more accurate and personalized response to the user request based on the task knowledge.

- We propose a Multi-Attention Large Language Model (MAS2S) architecture which considers task knowledge to generate clarification questions for user request and user profile on task-oriented information seeking.

- The experiment results of MAS2S achieve the best performances against baselines on general information seeking and task-oriented dialogue systems. Extensive experimental analysis also helps to better understand the advantages of our method.

5.2 Problem Formalization

5.2.1 The Ask Paradigm in Task-Oriented Dialogue Systems

Task-oriented information-seeking is an important problem that needs to be solved by task-oriented dialogue systems. A key advantage of dialogue systems is that the system can ask questions from users actively, so as to understand users' information needs accurately, and to increase its confidence in the information-seeking results. Based on this philosophy, we propose a clarification question generation paradigm for information seeking on task-oriented dialogues, as shown in Figure 5.2, named The Ask Paradigm in Task-Oriented Dialogue Systems.

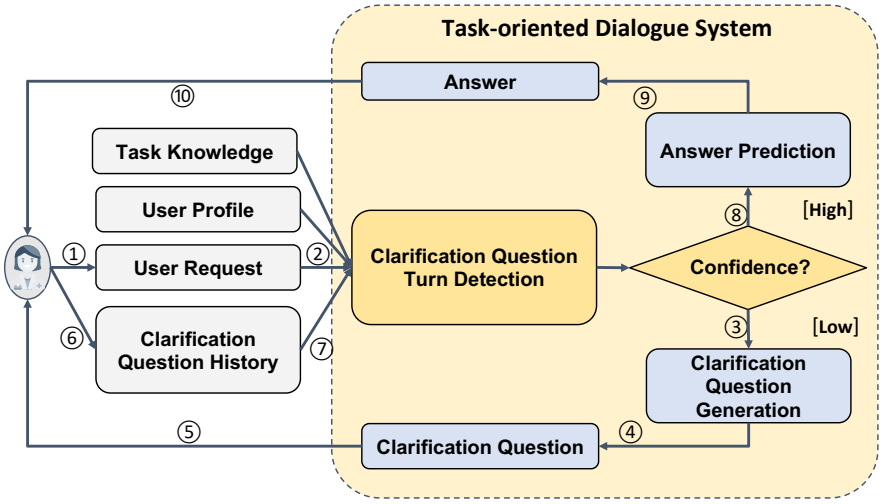


Figure 5.2: The Ask Paradigm in Task-Oriented Dialogue Systems.

After a user provides an initial user request related to a task, the system calculates the answer confidence with the *clarification question turn detection* module based on the user request, the user profile, and the task knowledge. If system is not sufficiently confident with the answer, it will generate a clarification question to ask using the *clarification question generation* module, which considers user request, user profile, and task knowledge.

After the user gives feedback to the clarification question, the system enters the loop again, but this time it considers not only the users initial request, but also the newly collected clarification question and user feedback which contains new information about the user's needs and user profiles. This process will continue until the system is confident enough to provide an answer, in which case the system will display the answer to the user and the conversation will stop.

5.2.2 Notations and Problem Statement

When a user wants to seek information about a task, the user will first give an initial user request R that describes the user's information needs. This user also has a user profile U that describes the personalized background information. If the user request R and user profile U are underspecified, the system cannot provide an accurate and personalized answer Y to the user request R. Therefore, the system needs to use the task knowledge T to infer a clarification question Q to clarify the user request R and user profile U. We build the following conversation for this task-oriented information-seeking,

$$R, U, T|Q_1, A_1, Q_2, A_2, ..., Q_K, A_K|Y, \tag{5.2.1}$$

where Q_k is a clarification question asked by the system, A_k is the user answer to the question Q_k, and K is the number of clarification questions in the dialogue.

Based on the above notation, the task-oriented dialogue system aims at learning models for the following two tasks:

Clarification Question Generation. Given a user request, a user profile, a task knowledge, and a dialogue history, generate the next clarification question

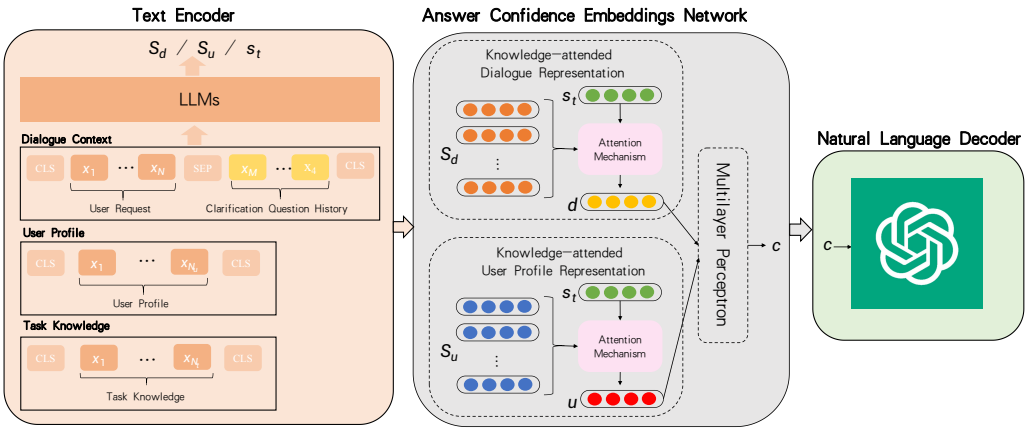


Figure 5.3: The architecture of MAS2S for information seeking on task-oriented dialogues.

to ask. Specifically, a generative model is trained by maximizing the probability of each clarification question in each of the dialogues:

$$P(Q_k|R, U, T, Q_1, A_1, \ldots, Q_{k-1}, A_{k-1}), \quad k \in \{1, \ldots, K\} \quad (5.2.2)$$

Answer Prediction. Given a user request, a user profile, a task knowledge, and a dialogue history, generate the answer for the user request. Specifically, a generation model is trained by maximizing the probability of the ground truth result for each of the dialogues:

$$P(Y|R, U, T, Q_1, A_1, ..., Q_K, A_K) \quad (5.2.3)$$

5.3 Multi-Attention Large Language Model

In this section, we propose a Multi-Attention Large language Model framework, named MALLM, that is able to ask clarification questions based on the Task-oriented System Ask paradigm. MAS2S formalizes clarification question generation and answer prediction as a sequence-to-sequence problem using LLMs [192] and Attention Networks [182]. As shown in Figure 7.2, MAS2S consists of a text encoder, an answer confidence embedding network, and a natural language decoder. At each turn of the dialogue, the text encoder transforms the user request and the clarification question history into dialogue embeddings using the LLMs; the text encoder transforms the user profile into the user embeddings using the LLMs; the text encoder transforms the task knowledge into the knowledge embeddings also using the LLMs; the answer confidence embeddings network creates knowledge-aware dialogue representations and knowledge-aware user representations using the attention mechanism to calculate answer confidence embeddings; finally, the natural language decoder sequentially generates a clarification question or an answer for the user request on the basis of the answer confidence embeddings. Below we describe MAS2S in detail.

5.3.1 Text Encoder

The text encoder takes the text of a dialogue, a user profile, and of a task knowledge as input respectively and employs LLMs to construct the corresponding semantic embeddings.

More specifically, to generate the semantic embeddings of dialogue context, the LLMs is given the token sequence:

$X = ([\text{CLS}], x_1, ..., x_N, [\text{SEP}], y_1, ..., y_M, [\text{CLS}])$,

which are the sub-word tokens of user request with length N and the clarification question history with length M. The [CLS] and [SEP] are the start-of-text/end-of-text and separator pseudo-tokens. The output embeddings of each token are used as the dialogue semantic embeddings, referred to as $S_d = (d_1, ..., d_{N+M+3})$.

To generate the semantic embeddings of a user profile, the LLMs takes a sequence of user profile tokens with length N_u as inputs, denoted as $X = ([\text{CLS}], x_1, ..., x_{N_u}, [\text{CLS}])$. The output is a sequence of embeddings with length $N_u + 2$, denoted as $S_u = (u_1, ..., u_{N_u+2})$ and referred to as user profile embeddings, with one embedding for each token.

We also use LLMs to generate representations for task knowledge. The input is a sequence of task knowledge tokens with length N_t, denoted as $X = ([\text{CLS}], x_1, ..., x_{N_t}, [\text{CLS}])$. The state of the final [CLS] is used as the task knowledge semantic embeddings s_t.

5.3.2 Answer Confidence Embeddings Network

The answer confidence embeddings network takes the sequence of dialogue embeddings, the sequence of user profile embeddings, and the task knowledge embeddings as input and first calculates knowledge-attended dialogue representations, and the knowledge-attended user profile representations. In this way, the semantic information from dialogue context and user profile is represented based on the task knowledge. Then the answer confidence embeddings can be obtained by the reconstructed knowledge-attended semantic embeddings.

Specifically, we first use the attention mechanism to calculate the knowledge-attended representations between task knowledge s_t and the dialogue S_d / user profile S_u by bilinear interaction:

$$A_d = \text{softmax}(\exp(S_d^T W_d s_t)) \tag{5.3.1}$$

$$A_u = \text{softmax}(\exp(S_u^T W_u s_t)), \tag{5.3.2}$$

where W_d and W_u are the bilinear interaction matrices to be learned. Then the knowledge-attended dialogue representations d and the knowledge-attended user profile representations u are calculated as $d = S_d^T A_d$ and $u = S_u^T A_u$.

To obtain the answer confidence embedding c for current dialogues and users, we concatenate the knowledge-attended dialogue representations and the knowledge-attended user profile representations. A multi-layer perceptron derives the answer confidence embedding c by the following equation:

$$c = \text{MLP}([d; u]). \tag{5.3.3}$$

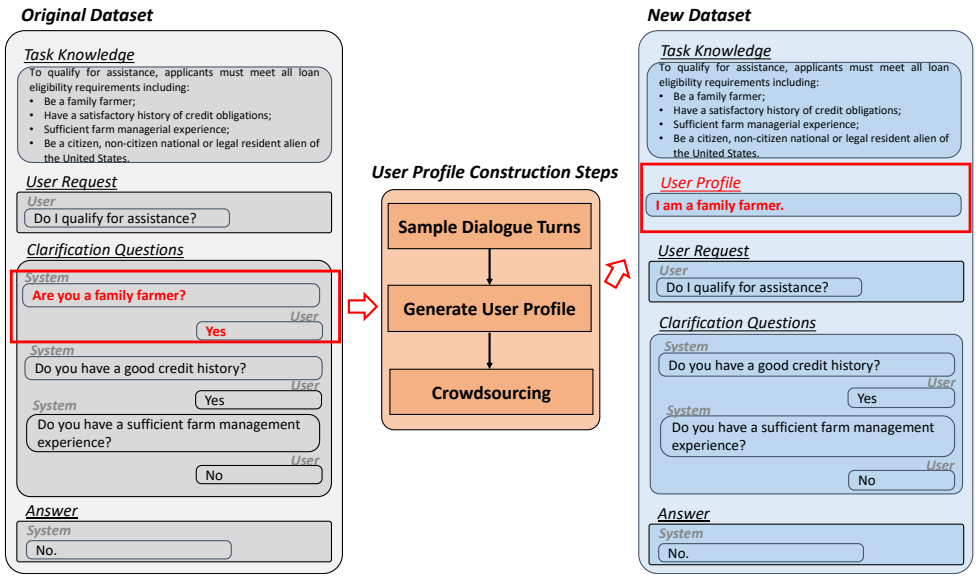


Figure 5.4: The process of user profile construction. We follow a three-step strategy as follows: (1) Sample dialogue turns; (2) Generate user profile; and (3) Correct grammatical errors through crowdsourcing.

5.3.3 Natural Language Decoder

The natural language decoder generates clarification questions or answers to the user's request by attending to the answer confidence embeddings. We employ LLMs for the natural language decoder, which takes the answer confidence embedding c as its initial input embedding.

5.3.4 Training

The training follows the standard procedure of a sequence-to-sequence model. The Answer Confidence Embeddings Network is fine-tuned in the training process. Cross-entropy loss is utilized to measure the loss of generating clarification questions and answers.

5.4 Data Collection

In task-oriented information seeking, to provide accurate and personalized answers, the system needs to clarify user requests and user profiles based on task knowledge. To the best of our knowledge, no datasets have been developed for this purpose. The closest dataset that can be used is the ShARC[193] dataset, which asks clarification questions based on task knowledge to clarify user information-seeking requests. The left example in Figure 5.4 is from the ShARC dataset. However, the ShARC dataset does not contain user profiles, which can not be directly used in our settings. To build a suitable dataset for our setting, we need to extend each dialogue in ShARC dataset with a user profile. The new extended dataset is called ClariT. This is the first public dataset that focuses on asking clarification questions for information seeking on task-oriented dialogues. In this section, we explain how we extend the ShARC dataset for task-oriented information seeking.

Table 5.1: Number of dialogues, task knowledge, user profiles, and turns in ClariT.

Set	#Dialogue	#Task Knowledge	#User Profile	#Turns
All	108,599	1,742	85,749	260,924
Training	76,019	687	55,048	184,027
Validation	10,860	495	10,545	25,473
Testing	21,720	560	20,156	51,424

5.4.1 User Profile Construction

Because the original ShARC dataset lacks personalized information, we need to construct a user profile for each dialogue. Many researchers have demonstrated that dialogue context usually contains strong suggestions for personalized information [194, 195]. Therefore, following previous work [196, 197], we also utilize dialogue context to construct user profiles. As shown in Figure 5.4, we use a three-step strategy to construct user profiles. In the first step, we sample some dialogue context. In the second step, we rewrite the sampled dialogue context into declarative sentences as the user profile using rules. In the third step, we use human annotators to correct any grammatical errors incurred during the rewriting process.

Specifically, we first randomly sample 1-5 dialogue turns from the dialogue context. Each dialogue turn in the original dialogue context consists of a clarification question and a corresponding answer. We use the sampled clarification question and answer pairs to construct the user profile. Given the generated user profile will contain the answer to these clarification questions, these clarification questions are unnecessary to be asked in the dialogue. Therefore, we remove these clarification questions from the original dialogue context to improve the efficiency of the dialogue.

After we sample dialogue turns, we need to rewrite them into declarative sentences to generate user profiles. For each dialogue turn, we identify the auxiliary verb in the clarification questions (such as “Are”, “Do”, etc.) and the polarity of the answers (such as “Yes” and “No”). Based on the auxiliary verb and polarity, we use rules to map each clarification question and answer pair into a declarative sentence. For example, in Figure 5.4, the clarification question and answer pair is “Are you a family farmer? Yes”. The rewritten sentence is “I am a family farmer.” Finally, the user profile is constructed by concatenating all these rewritten declarative sentences.

However, this rewriting process is not always as straightforward; in these cases, to improve the quality of the generated user profiles, we collect human annotations to correct grammatical errors incurred during the rewriting process. It is important to note that our annotators not only check grammatical errors in the rewriting declarative sentences but also are required to provide suitable corrections for the errors as well. All the grammatical errors in the rewriting of declarative sentences are replaced with the corrections provided by the annotators.

After that, we have a new dataset ClariT that is suitable for task-oriented information-seeking problems. We split ClariT into train, development, and test sets such that the train set includes 70% of the conversations, the development set contains 10% of them, and the rest 20% is the test set. Details about the ClariT dataset are shown in Table 5.1.

5.4.2 Dataset Quality Check

We conduct a further human evaluation to assess the quality of ClariT. Following previous work [184], three annotators were asked to evaluate the quality of ClariT. The criterion for dataset quality evaluation contains five dimensions: 1) Fluency: Is the user profile grammatically well-formed? 2) Usefulness: Does the user profile have useful personalized information? 3) Relevancy: Is the dialogue context relevant to user request and user profile? 4) Clarification: Does the dialogue context clarify unclear information in the user request and user profile? 5) Naturalness: Since we removed the sampled dialogue turns from the dialogue context, the naturalness of the dialogues in ClariT may be worse than the original dialogues. Therefore, we conducted a comparative study where we show annotators one dialogue from ClariT and one original dialogue, by asking annotators to identify which dialogue is more natural.

We randomly sampled 100 dialogues from ClariT. Under fluency, usefulness, relevancy, and clarification dimensions, the ratios of dialogues that are satisfied with the corresponding dimension are all 1.0. For the naturalness dimension, the ratio of dialogues identified as more natural by ClariT is approximately 0.5. This indicates that ClariT is as natural as the original ShARC dataset. The evaluation results on relevancy, usefulness, fluency, clarification, and naturalness indicate the high quality of ClariT.

5.5 Experiments

5.5.1 Experiments Setup

Dataset. We conduct the experiments using the benchmark datasets on task-oriented information seeking, namely ClariT, which was built using ShARC[193] dataset. More details of the datasets are shown in Table 5.1.

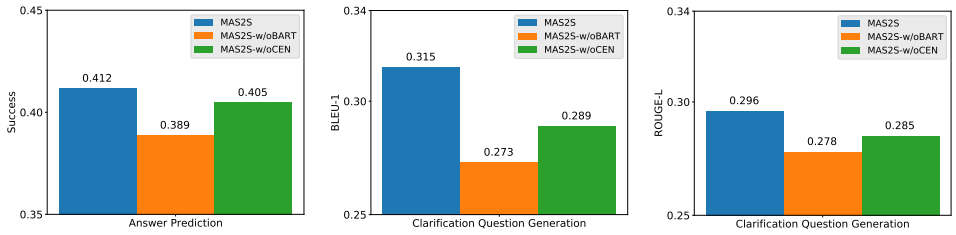
Baselines. We compare our approach with the following state-of-the-art baselines, which include clarification question generation methods and task-oriented dialogue system response generation methods. Since the baselines cannot utilize task knowledge and user profile information, to make the comparison between MAS2S and baselines fair, we concatenate the task knowledge, user profile, and dialogue context together as the inputs of baselines.

GAN-Utility [89]: State-of-the-art on clarification question generation, which is a sequence-to-sequence generative network for generating clarification questions in open-domain dialogues. We reproduce the baseline using their open-source code.

SOLOIST [198]: State-of-the-art on task-oriented dialogue system response generation, which uses a transformer-based auto-regressive language model to generate system responses. Since we don't have the dialogue states and dialogue actions in ClariT, this model is only trained on the loss of dialogue response generation. We reproduce the baseline using their open-source code.

UBAR [199]: State-of-the-art on task-oriented dialogue system response generation, which utilizes the large pre-trained language model to generate system responses on the sequence of the entire task-oriented dialogue session. Similar to SOLOIST, this model is only trained on the loss of dialogue system response generation. We reproduce the baseline using their open-source code.

Evaluation Measures. For the evaluation of *Clarification Question Generation*,



(a) Success (b) BLUE (c) ROUGE

Figure 5.5: Ablation study of MAS2S with respect to LLMs, and confidence embeddings network on ClariT.

we follow the previous work [89, 198, 199, 200], using both automatic metrics and human evaluation. Specifically, we estimate a generated response by automatically measuring its gram-based accuracy against the ground truth. The automatic metrics we used are as follow: (1) *BLEU* [201] estimates a generated clarification question via measuring its n-gram precision against the ground truth; (2) *ROUGE* [202] measures n-gram recall between generated clarification question and ground truth.

For the evaluation of *Answer Prediction*, we follow the previous work [89, 200], using *Success* metric. *Success* is calculated by measuring how often the dialogue system provides the right answer to the user request.

A paired t-test is conducted to determine whether there is a significant difference between our method and the best baseline. We compare our method with UBAR. The significance level (p-value) is set at 0.05.

Implementation. We use a pre-trained $llama - 7b$ model to encode dialogue context, user profiles, and task knowledge, and decode the system response. The answer confidence embeddings Network is trained in the training process. The hidden size of all the attention layers is set to 768. The dropout probability is set to 0.1. The batch size is set to 4. We optimize with Adam optimizer [178] and an initial learning rate of 1e-4. Hyperparameters are chosen using the validation dataset in all cases.

5.5.2 Experimental Results

For the clarification question generation task, the automatic metric evaluation results on ClariT are shown in Table 5.2. We can see that MAS2S performs significantly better than the baselines. The results indicate that MAS2S can effectively leverage the task knowledge to clarify user requests and user profiles for task-oriented information seeking. MAS2S, which contains clarification question turn detection module, is able to produce much more useful clarification questions.

For the answer prediction task, the experiment results on ClariT are shown in Table 5.2. We can observe that MAS2S performs significantly better than the baselines in terms of Success. The results indicate that MAS2S, which contains the task knowledge attended answer confidence embedding network, is able to provide much more accurate answers for task-oriented information seeking.

Table 5.2: Performance of MAS2S and baselines on automatic metrics. Numbers in **bold** denote best results in that metric. MAS2S significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$).

Model	Success	BLEU	ROUGE
GAN-Utility	0.334	0.224	0.216
SOLOIST	0.352	0.229	0.219
UBAR	0.397	0.282	0.273
MAS2S	**0.453**	**0.327**	**0.342**

5.6 Discussions

We seek to answer the following research questions in our experiments: (Q1) What are the effects of different components in MAS2S? (Q2) What are the key factors that contribute to the high performance of MAS2S? (Q3) Do the generated clarification questions have the utility to provide the right answer to the user request? (Q4) Are the generated clarification questions necessary? (Q5) What is the impact of the user request length on the performance? (Q6) What is the impact of the user profile on the performance?

5.6.1 Ablation Study

To answer the question (Q1) "What are the effects of different components in MAS2S?", we conduct an ablation study on MAS2S. We validate the effects of two factors: confidence embeddings network and LLMs-based encoder/decoder. The results indicate that all the components of MAS2S are indispensable.

Effect of Confidence Embeddings Network. To investigate the effectiveness of using the confidence embeddings network, we compare MAS2S with MAS2S-w/oCEN, which eliminates the confidence embeddings network module. We concatenate the semantic embeddings of dialogue context, user profile, and task knowledge as the initial state of the decoder. Figure 6.2 shows the results on ClariT in terms of BLEU-1, ROUGE-L, and Success. From the results, we can see that the performances of both clarification question generation and answer prediction deteriorate considerably without the confidence embeddings network. This indicates that the confidence embeddings network helps provide a more accurate indication about when and how to ask clarification questions and predict answers.

Effect of LLMs. To investigate the effectiveness of using LLMs in the text encoder, and natural language decoder, we replace LLMs with the standard sequence-to-sequence Transformer [182] and run the model on ClariT. As shown in Figure 6.2, the performances on both clarification question generation and answer prediction of the Transformer-based model MAS2S-w/oLLMs decreases significantly compared with MAS2S, in terms of BLEU-1, ROUGE-L, and Success. It indicates that the LLMs-based encoder/decoder can create more accurate representations for dialogue, user profile, and task knowledge on both clarification question generation and answer prediction.

Task Knowledge	Dialogue Context	
Businesses that use Centrepay need to: 1. registration; 2. licensing and accreditation; 3. financial and privacy laws; 4. layby services.	• **User Request:** U: Can my Business use Centrepay? • **Clarification Question History:** S: Do you have licensing and accreditation? U: Yes S: Do you use layby services? U: Yes	
User Profile	**Generated System Response**	
I follow all financial and privacy laws. My child is not isolated from school. We live in Philadelphia.	*Ground Truth*	Does your business have registration?
	MAS2S	Do you register?
	UBAR	Do you follow financial and privacy laws?

(a) Example 1

Task Knowledge	Dialogue Context	
You'll get Cold Weather Payments if you get Universal Credit, and one of the following apply: 1. you get a limited capability for work amount; 2. you get the disabled child amount; 3. you have a child under 5 living with you.	• **User Request:** U: Can I get Cold Weather Payments? • **Clarification Question History:** S: Do you have a child under 5 living with you? U: No S: Do you get the disabled child amount in your claim? U: No	
User Profile	**Generated System Response**	
I am unemployed, but I can work.	*Ground Truth*	No
	MAS2S	No
	UBAR	Do you have the capability for work?

(b) Example 2

Figure 5.6: Case study based on MAS2S and the best baseline UBAR on ClariT. The generated system response in green is correctly predicted, while the generated system response in red is incorrectly predicted.

5.6.2 Case Study

To answer the question (Q2) "What are the key factors that contribute to the high performance of MAS2S?", we qualitatively analyze the results of MAS2S and the best baseline UBAR on ClariT dataset. We find that MAS2S generates more accurate system responses by leveraging the relation existing in the dialogue context, user profile and task knowledge. For example, in the first case in Figure 7.5, the user profile mentions that *"I follow all financial and privacy laws"*. MAS2S can correctly infer the system needs to ask another clarification question about *"registration"* instead of *"financial and privacy laws"* considering the task knowledge and dialogue context. In the second case, the user profile mentions that *"I can work"*. MAS2S can correctly provide the answer *"No"* instead of asking a clarification question about *"capability for work"* also considering the task knowledge and dialogue context. From the examples above, we can see that MAS2S can effectively extract the relation between dialogue context, user profile, and task knowledge, yielding correct system responses. In contrast, UBAR can not model these relations. Thus it cannot properly generate responses.

5.6.3 Utility of Clarification Questions

To answer the question (Q3) "Do the generated clarification questions have the utility to provide the right answer to the user request?", we compare the performance of *Success* of MAS2S with the best baseline UBAR on dialogues asking different number of clarification questions. Figure 5.7 shows the performance of *Success* of MAS2S and UBAR on dialogues with k clarification questions ($k \in \{1, 2, 3, 4, 5\}$). The results show that MAS2S performs better in answer prediction on all dialogues. It indicates the usefulness of the clarification questions generated by MAS2S. In addition, we can also see the performance of MAS2S improves as the dialogue advances to multiple clarification questions. It indicates that appropriate clarification questions can effectively clarify user request to provide more accurate answer. We conjecture that the system can understand users information needs and user profile more accurate with an increase in the number of clarification questions.

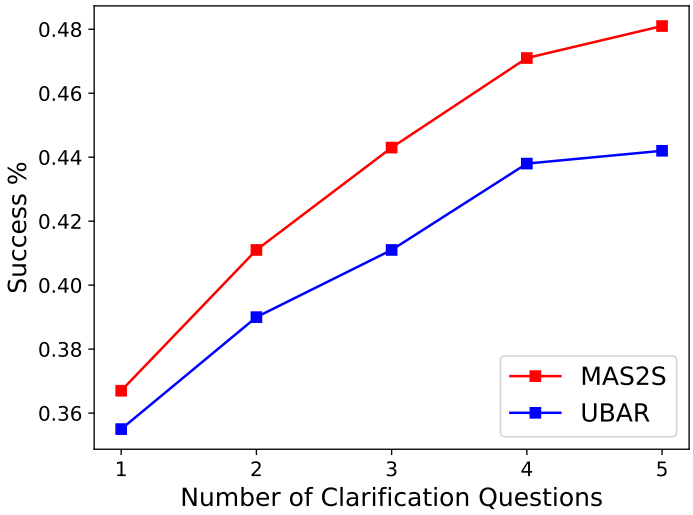


Figure 5.7: Impact of number of clarification questions on the performance of Success of MAS2S and UBAR.

5.6.4 Necessary of Clarification Questions

To answer the question (Q4) "Are the generated clarification questions necessary?", we compare MAS2S and the best baseline UBAR in terms of the average number of clarification questions (NoQ) over the dialogues, and the corresponding average absolute differences with the number of clarification questions needed (AbsDiff). As shown in Table 5.3, the average number of clarification questions (NoQ) of MAS2S is less than the best baseline UBAR. And when comparing average absolute differences with the number of clarification questions needed (AbsDiff), we observe that the number of clarification questions of MAS2S is more closer to the ground truth number of clarification questions (Oracle) than the best baseline UBAR. It indicates MAS2S can generate more necessary questions to clarify user request and user profile than UBAR.

Table 5.3: Average number of clarification questions (NoQ) and absolute difference of clarification questions (AbsDiff) over the dialogues of MAS2S and the best baseline UBAR on ClariT. **Bold** denotes best results.

Model	NoQ	AbsDiff
Oracle	2.36	0.00
UBAR	3.14	1.68
MAS2S	**2.97**	**0.94**

5.6.5 Impact of User Request Length

To answer the question (Q5) "What is the impact of the user request length on the performance of MAS2S?", we analyze the performance of MAS2S based on the number of user request tokens. Figure 5.8 shows the improved *Success* of MAS2S on different lengths of user request compared to the best baseline UBAR. From the results, one can observe that MAS2S performs better in all cases, no matter the length of the user request. It indicates that utilizing the clarification question generation model in task-oriented dialogue is necessary to clarify user request. In addition, the relative improvement of MAS2S is negatively correlated with the length of the user request. It indicates that the shorter user request needs clarification in more cases. We conjecture that it is due to shorter user request usually containing more ambiguous information. Asking clarification questions can effectively improve the natural language understanding ability of shorter user request in task-oriented dialogues.

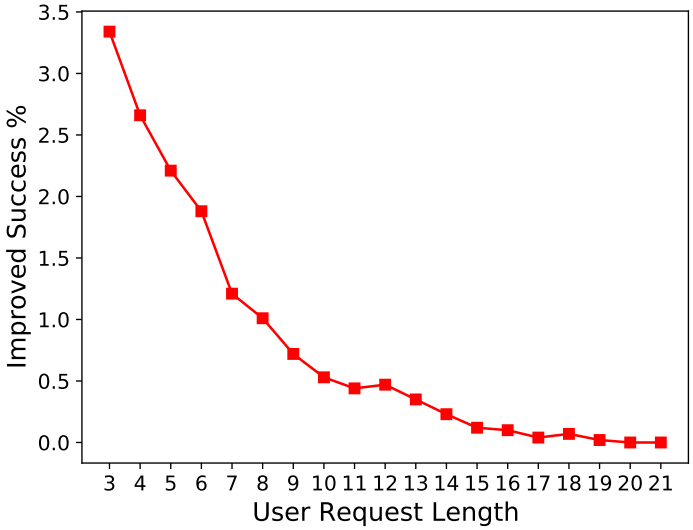


Figure 5.8: Impact of user request length on the performance of Success of MAS2S.

5.6.6 Impact of User Profile

To answer the question (Q6) "What is the impact of the user profile on the performance of MAS2S?", we compare MAS2S with MAS2S-w/oProfile, which eliminates the user profile and knowledge attended user profile representations.

Table 5.4 shows the results on ClariT in terms of *BLEU*, *ROUGE*, and *Success*. From the results, we can see that without user profiles, the performance of the clarification question generation deteriorates considerably. It indicates that the relation between the user profiles and the user requests helps the dialogue systems understand the user information needs. In addition, the performance of *Success* also deteriorates without using user profiles. We conjecture that it is due to the dialogue system's lack of personalization information to provide an accurate response to users. Thus, the utilization of user profiles is desirable.

Table 5.4: Performance of MAS2S and MAS2S-w/oProfile on ClariT dataset. Numbers in **bold** denote best results in that metric. MAS2S significantly improves over the MAS2S-w/oProfile (two-sided paired t-test, $p < 0.05$).

Model	Success	BLEU	ROUGE
MAS2S-w/oProfile	0.381	0.256	0.264
MAS2S	**0.412**	**0.315**	**0.296**

5.7 Summary

In this work, we focused on the problem of task-oriented information seeking. We formulate a new problem of asking clarifying questions for information seeking in task-oriented dialogues, which needs to clarify the user request and the user profile to provide a more accurate and personalized response based on the task knowledge. We proposed a Multi-Attention Seq2Seq Networks (MAS2S) to generate clarification questions and predict answers for task-oriented information seeking, which integrates the power of both sequential modeling and attention mechanisms. Experiments on ClariT verified the performance of MAS2S against state-of-the-art clarification question generation baselines and task-oriented dialogue system response generation baselines. The research on asking clarification questions for information seeking on task-oriented dialogues is still in its initial stage, and this work is just one of the first steps. In the future, the proposed paradigm may also be extended to more complex scenarios, such as considering task relations, etc.

Chapter 6

Schema-Guided User Satisfaction Modeling for Task-Oriented Dialogues

User Satisfaction Modeling (USM) is one of the popular choices for task-oriented dialogue systems evaluation, where user satisfaction typically depends on whether the user's task goals were fulfilled by the system. Task-oriented dialogue systems use task schema, which is a set of task attributes, to encode the user's task goals. Existing studies on USM neglect explicitly modeling the user's task goals fulfillment using the task schema. In this paper, we propose SG-USM, a novel schema-guided user satisfaction modeling framework. It explicitly models the degree to which the user's preferences regarding the task attributes are fulfilled by the system for predicting the user's satisfaction level. SG-USM employs a pre-trained language model for encoding dialogue context and task attributes. Further, it employs a fulfillment representation layer for learning how many task attributes have been fulfilled in the dialogue, an importance predictor component for calculating the importance of task attributes. Finally, it predicts the user satisfaction based on task attribute fulfillment and task attribute importance. Experimental results on benchmark datasets (i.e. MWOZ, SGD, ReDial, and JDDC) show that SG-USM consistently outperforms competitive existing methods. Our extensive analysis demonstrates that SG-USM can improve the interpretability of user satisfaction modeling, has good scalability as it can effectively deal with unseen tasks and can also effectively work in low-resource settings by leveraging unlabeled data.[1]

6.1 Introduction

Task-oriented dialogue systems have emerged for helping users to solve specific tasks efficiently [166]. Evaluation is a crucial part of the development process of such systems. Many of the standard automatic evaluation metrics, e.g. BLEU [100], ROUGE [203], have been shown to be ineffective in task-oriented dialogue evaluation [102, 101]. As a consequence, User Satisfaction Modeling (USM) [7, 204, 122, 120, 205] has gained momentum as the core evaluation metric for task-oriented dialogue systems. USM estimates the over-

[1] Code is available at `https://github.com/amzn/user-satisfaction-modeling`.

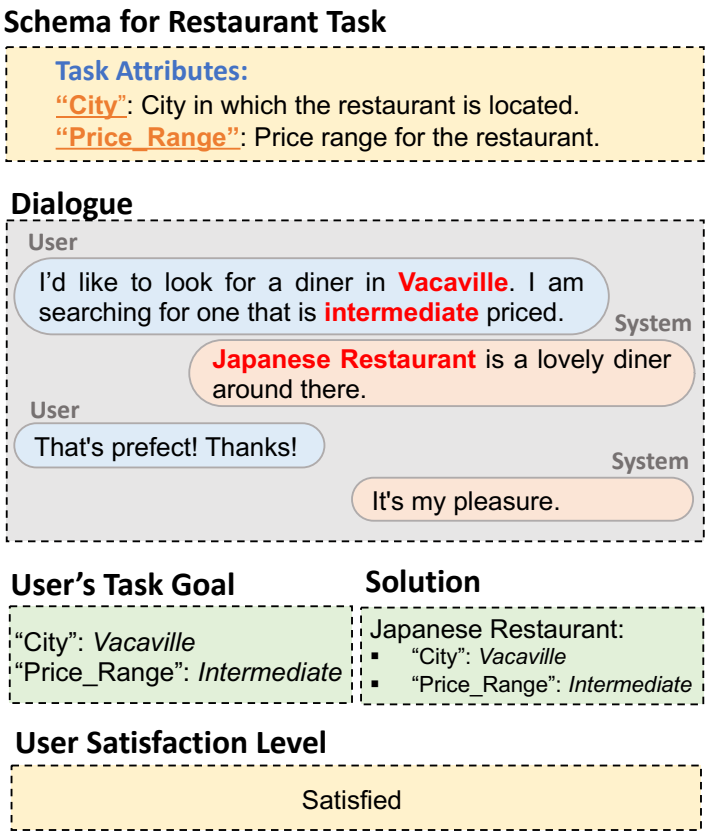


Figure 6.1: Task-oriented dialogue system has a predefined schema for each task, which is composed of a set of task attributes. In a dialogue, the user's task goal is encoded by the task attribute and value pairs. The user is satisfied with the service when the provided solution fulfills the user's preferences for the task attributes.

all satisfaction of a user interaction with the system. In task-oriented dialogue systems, whether a user is satisfied largely depends on how well the users task goals were fulfilled. Each task would typically have an associated task schema, which is a set of task attributes (e.g. location, date for check-in and check-out, etc. for a hotel booking task), and for the user to be satisfied, the system is expected to fulfill the user's preferences about these task attributes. Figure 7.1 shows an example of USM for task-oriented dialogues.

Effective USM models should have the following abilities: (1) Interpretability by giving insights on what aspect of the task the system performs well. For instance, this can help the system to recover from an error and optimize it toward an individual aspect to avoid dissatisfaction. (2) Scalability in dealing with unseen tasks, e.g. the model does not need to retrain when integrating new tasks. (3) Cost-efficiency for performing well in low-resource settings where it is often hard to collect and expensive to annotate task-specific data.

Previous work in USM follows two main lines of research. First, several methods use user behavior or system actions to model user satisfaction. In this setting, it is assumed that user satisfaction can be reflected by user behaviors or system actions in task-oriented dialogue systems, such as click, pause, request, inform [114, 115]. A second approach is to analyze semantic information in user natural language feedback to estimate user satisfaction, such as sentiment analysis [7, 120] or response quality assessment [122, 123]. However, both of

these two lines of work do not take into account the abilities of interpretability, scalability, and cost-efficiency.

In this paper, we propose a novel approach to USM, referred to as Schema-Guided User Satisfaction Modeling (SG-USM). We hypothesize that user satisfaction should be predicted by the fulfillment degree of the users task goals that are typically represented by a set of task attribute and value pairs. Therefore, we explicitly formalize this by predicting how many task attributes fulfill the user's preferences and how important these attributes are. When more important attributes are fulfilled, task-oriented dialogue systems should achieve better user satisfaction.

Specifically, SG-USM comprises a pre-trained text encoder to represent dialogue context and task attributes, a task attribute fulfillment representation layer to represent the fulfillment based on the relation between the dialogue context and task attributions, a task attribute importance predictor to calculate the importance based on the task attributes popularity in labeled and unlabeled dialogue corpus, and a user satisfaction predictor which uses task attributes fulfillment and task attributes importance to predict user satisfaction. SG-USM uses task attributes fulfillment and task attributes importance to explicitly model the fulfillment degree of the user's task goals (interpetability). It uses an task-agnostic text encoder to create representations of task attributes by description, no matter whether the task are seen or not (scalability). Finally, it uses unlabeled dialogues in low-resource settings (cost-efficiency).

Experimental results on popular task-oriented benchmark datasets show that SG-SUM substantially and consistently outperforms existing methods on user satisfaction modeling. Extensive analysis also reveals the significance of explicitly modeling the fulfillment degree of the user's task goals, the ability to deal with unseen tasks, and the effectiveness of utilizing unlabeled dialogues.

6.2 Schema-guided User Satisfaction Modeling

Our SG-USM approach formalizes user satisfaction modeling by representing the user's task goals as a set of task attributes, as shown in Figure 7.1. The goal is to explicitly model the degree to which task attributes are fulfilled, taking into account the importance of the attributes. As shown in Figure 7.2, SG-USM consists of a text encoder, a task attribute fulfillment representation layer, a task attribute importance predictor, and a user satisfaction predictor. Specifically, the text encoder transforms dialogue context and task attributes into dialogue embeddings and task attribute embeddings using BERT [22]. The task attribute fulfillment representation layer models relations between the dialogue embeddings and the task attribute embeddings by attention mechanism to create task attribute fulfillment representations. Further, the task attribute importance predictor models the task attribute popularity in labeled and unlabeled dialogues by the ranking model to obtain task attribute importance weights. Finally, the user satisfaction predictor predicts user satisfaction score on the basis of the task attribute fulfillment representations and task attribute importance weights using a multilayer perceptron.

6.2.1 Text Encoder

The text encoder takes the dialogue context (user and system utterances) and the descriptions of task attributes as input and uses BERT to obtain dialogue and task attribute embeddings, respectively.

Considering the limitation of the maximum input sequence length of BERT, we encode dialogue context by each dialogue turn. Specifically, the BERT encoder takes as input a sequence of tokens with length L, denoted as $X = (x_1, ..., x_L)$. The first token x_1 is [CLS], followed by the tokens of the user utterance and the tokens of the system utterance in one dialogue turn, separated by [SEP]. The representation of [CLS] is used as the embedding of the dialogue turn. Given a dialogue with N dialogue turns, the output dialogue embeddings is the concatenation of all dialogue turn embeddings $D = [d_1; d_2; ...; d_N]$.

To obtain task attribute embeddings, the input is a sequence of tokens with length K, denoted as $Y = \{y_1, ..., y_K\}$. The sequence starts with [CLS], followed by the tokens of the task attribute description. The representation of [CLS] is used as the embedding of the task attribute. The set of task attribute embeddings are denoted as $T = \{t_1, t_2, ..., t_M\}$, where M is the number of task attributes.

6.2.2 Task Attribute Fulfillment Representation Layer

The task attribute fulfillment representation layer takes the dialogue and task attribute embeddings as input and calculates dialogue-attended task attribute fulfillment representations. This way, whether each task attribute can be fulfilled in the dialogue context is represented.

Specifically, the task attribute fulfillment representation layer constructs an attention vector by a bilinear interaction, indicating the relevance between dialogue and task attribute embeddings. Given the dialogue embeddings D and i-th task attribute embedding t_i , it calculates the relevance as follows,

$$A_i = \text{softmax}(\exp(D^T W_a t_i)), \tag{6.2.1}$$

where W_a is the bilinear interaction matrix to be learned. A_i represents the attention weights of dialogue turns with respect to the i-th task attribute. Then the dialogue-attended i-th task attribute fulfillment representations are calculated as follows,

$$t_i^a = DA_i. \tag{6.2.2}$$

The dialogue-attended task attribute fulfillment representations for all task attributes are denoted as:

$$T^a = [t_1^a, t_2^a, ..., t_M^a]. \tag{6.2.3}$$

where M is the number of the task attributes.

6.2.3 Task Attribute Importance Predictor

The task attribute importance predictor also takes the dialogue and task attribute embeddings as input and calculates attribute importance scores. The importance scores are obtained by considering both the task attribute presence frequency and task attribute presence position in the dialogue.

First, we use the Maximal Marginal Relevance (MMR) [206] to select the top relevant task attributes for the dialogue context. The selected task attributes are then used to calculate the task attribute presence frequency in the dialogue. The MMR takes the j-th dialogue turn embeddings d_j and task attribute embeddings T as input, and picks the top K relevant task attributes for the j-th dialogue turn:

$$R_j = \underset{t_i \in T \setminus U}{\operatorname{argmax}} [\lambda \cos(t_i, d_j) - (1 - \lambda) \max_{t_k \in U} \cos(t_i, t_k)] \tag{6.2.4}$$

where U is the subset of attributes already selected as top relevant task attributes, cos() is the cosine similarity between the embeddings. λ trades off between the similarity of the selected task attributes to the dialogue turn and also controls the diversity among the selected task attributes. The task attribute presence frequency vector for the j-th dialogue turn is computed as follows,

$$F_j = [f_j^1, f_j^2, f_j^3, ..., f_j^M] \tag{6.2.5}$$

$$f_j^i = \begin{cases} 1 & i \in R_j \\ 0 & i \notin R_j \end{cases} \tag{6.2.6}$$

where M is the number of the task attributes.

However, the task attribute presence frequency vector does not reward task attributes that appear in the beginning of the dialogue. The premise of task attribute importance score is that task attributes appearing near the end of the dialogue should be penalized as the graded importance value is reduced logarithmically proportional to the position of the dialogue turn. A common effective discounting method is to divide by the natural log of the position:

$$\widetilde{F_j} = \frac{F_j}{log(j+1)} \tag{6.2.7}$$

The task attribute importance predictor then computes the importance score on the basis of the sum of the discounted task attribute presence frequency of all dialogues. Given the dialogue corpus (including both labeled and unlabeled dialogues) with Z dialogues $C = \{D_1, D_2, ..., D_Z\}$, the task attribute importance scores are calculated as follow:

$$S = \text{softmax}(\sum_{l=1}^{Z} \sum_{j=1}^{\text{Num}(D_l)} \widetilde{F_j^l}) \tag{6.2.8}$$

where Num() is the number of the dialogue turn in dialogue D_l, and $\widetilde{F_j^l}$ is the discounted task attribute presence frequency of j-th dialogue turn in dialogue D_l.

6.2.4 User Satisfaction Predictor

Given the dialogue-attended task attribute fulfillment representations T^a and the task attribute importance scores S, the user satisfaction labels are obtained by aggregating task attribute fulfillment representations based on task attribute importance scores. This way, the user satisfaction is explicitly modeled by the fulfillment of the task attributes and their individual importance.

Specifically, an aggregation layer integrates the dialogue-attended task attribute fulfillment representations by the task attribute importance scores as follows:

$$h = T^a S \tag{6.2.9}$$

Then the Multilayer Perceptron (MLP) [207] with softmax normalization is employed to calculate the probability distribution of user satisfaction classes:

$$p = \text{softmax}(\text{MLP}(h)) \tag{6.2.10}$$

6.2.5 Training

We train SG-USM in an end-to-end fashion by minimizing the cross-entropy loss between the predicted user satisfaction probabilities and the ground-truth satisfaction:

$$\mathcal{L} = -y\log(p) \tag{6.2.11}$$

where y is the ground-truth user satisfaction. Pre-trained BERT encoders are used for encoding representations of utterances and schema descriptions respectively. The encoders are fine-tuned during the training process.

6.3 Experimental Setup

6.3.1 Datasets

We conduct experiments using four benchmark datasets containing task-oriented dialogue on different domains and languages (English and Chinese), including MultiWOZ2.1 (MWOZ) [91], Schema Guided Dialogue (SGD) [99], ReDial [137], and JDDC [208].

MWOZ and SGD are English multi-domain task-oriented dialogue datasets, which include hotel, restaurant, flight, etc. These datasets contain domain-slot pairs, where the slot information could correspond to the task attributes.

ReDial is an English conversational recommendation dataset for movie recommendation. The task attributes are obtained from the Movie[2] type on Schema.org.

JDDC is a Chinese customer service dialogue dataset in E-Commerce. The task attributes are obtained from the Product[3] type on Schema.org.cn, which provides schemas in Chinese.

Specifically, we use the subsets of these datasets with the user satisfaction annotation for evaluation, which is provided by Sun et al [7]. We also use the subsets of these datasets without the user satisfaction annotation to investigate the semi-supervised learning abilities of SG-USM. Table 6.1 displays the statistics of the datasets in the experiments.

[2]https://schema.org/Movie

[3]https://schema.org.cn/Product

Characteristics	MWOZ	SGD	ReDial	JDDC
Language	English	English	English	Chinese
#Dialogues	1,000	1,000	1,000	3,300
#Utterances	12,553	13,833	11,806	54,517
#Avg Turn	23.1	26.7	22.5	32.3
#Attributes	37	215	128	13
%Sat. Class	27:39:34	22:30:48	23:26:51	23:53:24
#TrainSplit	7,648	8,674	7,372	38,146
#ValidSplit	952	1,074	700	5,006
#TestSplit	953	1,085	547	4,765
#Unlabeled Dialogues	4,000	4,000	4,000	4,000

Table 6.1: Statistics of the task-oriented dialogue datasets.

Model	MWOZ				SGD				ReDial				JDDC			
	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
HiGRU	44.6	43.7	44.3	43.7	50.0	47.3	48.4	47.5	46.1	44.4	44.0	43.5	59.7	57.3	50.4	52.0
HAN	39.0	37.1	37.1	36.8	47.7	47.1	44.8	44.9	46.3	40.0	40.3	40.0	58.4	54.2	50.1	51.2
Transformer	42.8	41.5	41.9	41.7	53.1	48.3	49.9	49.1	47.5	44.9	44.7	44.8	60.9	59.2	53.4	56.2
BERT	46.1	45.5	47.4	45.9	56.2	55.0	53.7	53.7	53.6	50.5	51.3	50.0	60.4	59.8	58.8	59.5
USDA	49.9	49.2	49.0	48.9	61.4	60.1	55.7	57.0	57.3	54.3	52.9	53.4	61.8	62.8	63.7	61.7
SG-USM-L	50.8*	49.3	50.2*	49.4*	62.6*	58.5	57.2*	57.8*	57.9*	54.7	53.0	53.8	62.5*	62.6	63.9	62.8*
SG-USM-L&U	**52.3***	**50.4***	**51.4***	**50.9***	**64.7***	**61.6***	**58.8***	**60.2***	**58.4***	**55.8***	**53.2***	**54.5***	**63.3***	**63.1***	**64.1***	**63.5***

Table 6.2: Performance of SG-USM and baselines on various evaluation benchmarks. Numbers in **bold** denote the best model performance for a given metric. Numbers with * indicate that SG-USM model is better than the best-performing baseline method (underlined scores) with statistical significance (t-test, $p < 0.05$).

6.3.2 Baselines and SG-USM Variants

We compare our SG-USM approach with competitive baselines as well as state-of-the-art methods in user satisfaction modeling.

HiGRU [209] proposes a hierarchical structure to encode each turn in the dialogue using a word-level gated recurrent unit (GRU) [210] and a sentence-level GRU. It uses the last hidden states of the sentence-level GRU as inputs of a multilayer perceptron (MLP) [207] to predict the user satisfaction level. We reproduce the baselines using their open-source code.

HAN [211] applies a two-level attention mechanism in the hierarchical structure of HiGRU to represent dialogues. An MLP takes the dialogue representation as inputs to predict the user satisfaction level. We reproduce the baselines using their open-source code.

Transformer [182] is a simple baseline that takes the dialogue context as input and uses the standard Transformer encoder to obtain the dialogue representations. An MLP is used on the encoder to predict the user satisfaction level. We reproduce the baselines using their open-source code.

BERT [22] concatenates the last 512 tokens of the dialogue context into a long sequence with a [SEP] token for separating dialogue turns. It uses the [CLS] token of a pre-trained BERT models to represent dialogues. An MLP is used on the BERT to predict the user satisfaction level. We reproduce the baselines using their open-source code.

USDA [114] employs a hierarchical BERT encoder to encode the whole dialogue context at the turn-level and the dialogue-level. It also incorporates the sequential dynamics of dialogue acts with the dialogue context in a multi-task framework for user satisfaction modeling. We reproduce the baselines using their open-source code.

We also report the performance of two simpler SG-USM variants:
SG-USM(L) only uses the dialogues with ground-truth user satisfaction labels to train the model.
SG-USM(L&U) uses both labeled and unlabeled dialogues in the training process. It takes the dialogues without user satisfaction annotation as the inputs of task attribute importance predictor module to obtain more general and accurate task attribute importance scores.

For a fair comparison with previous work and without loss of generality, we adopt BERT as the backbone encoder for all methods that use pre-trained language models.

6.3.3 Evaluation Metrics

Following previous work [114, 212, 213, 120], we consider a three-class classification task for user satisfaction modeling by treating the rating "</=/> 3" as "dissatisfied/neutral/satisfied". Accuracy (Acc), Precision (P), Recall (R), and F1 are used as the evaluation metrics.

A paired t-test is conducted to determine whether there is a significant difference between our method and the best baseline. We compare our method with USDA. The significance level (p-value) is set at 0.05.

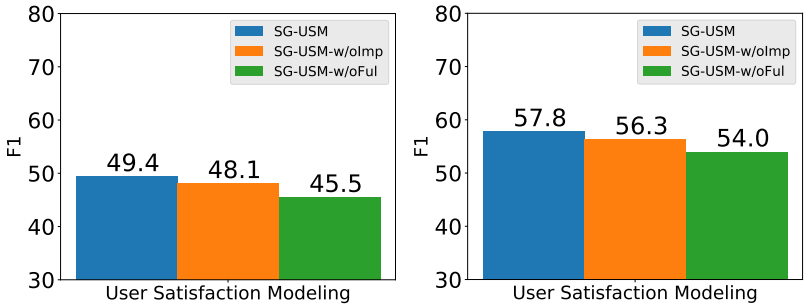
6.3.4 Training

We use BERT-Base uncased, which has 12 hidden layers of 768 units and 12 self-attention heads to encode the utterances and schema descriptions. We apply a two-layer MLP with the hidden size as 768 on top of the text encoders. ReLU is used as the activation function. The dropout probability is 0.1. Adam [178] is used for optimization with an initial learning rate of 1e-4. We train up to 20 epochs with a batch size of 16, and select the best checkpoints based on the F1 score on the validation set.

6.4 Experimental Results

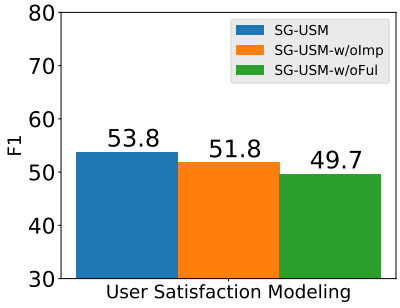
6.4.1 Overall Performance

Table 6.2 shows the results of SG-USM on MWOZ, SGD, ReDial, and JDDC datasets. Overall, we observe that SG-USM substantially and consistently outperforms all other methods across four datasets with a noticeable margin. Specifically, SG-USM(L) improves the performance of user satisfaction modeling via explicitly modeling the degree to which the task attributes are fulfilled. SG-USM(L&U) further aids the user satisfaction modeling via predicting task attribute importance based on both labeled dialogues and unlabeled dialogues. It appears that the success of SG-USM is due to its architecture design which consists of the task attribute fulfillment representation layer and the task attribute importance predictor. In addition, SG-USM can also effectively leverage unlabeled dialogues to alleviate the cost of user satisfaction score annotation.

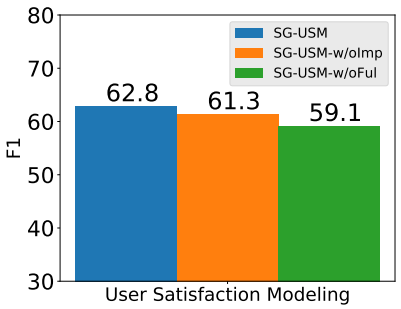


(a) MWOZ

(b) SGD



(c) ReDial



(d) JDDC

Figure 6.2: Performance of SG-USM by ablating the task attribute importance and task attribute fulfillment components across datasets.

6.4.2 Ablation Study

We also conduct an ablation study on SG-USM to study the contribution of its two main components: task attribute importance and task attribute fulfillment.

Effect of Task Attribute Importance

To investigate the effectiveness of task attribute importance in user satisfaction modeling, we eliminate the task attribute importance predictor and run the model on MWOZ, SGD, ReDial, and JDDC. As shown in Figure 6.2, the performance of SG-USM-w/oImp decreases substantially compared with SG-USM. This indicates that the task attribute importance is essential for user satisfaction modeling. We conjecture that it is due to the user satisfaction relates to the importance of the fulfilled task attributes.

Effect of Task Attribute Fulfillment

To investigate the effectiveness of task attribute fulfillment in user satisfaction modeling, we compare SG-USM with SG-USM-w/oFul which eliminates the task attribute fulfillment representation. Figure 6.2 shows the results on MWOZ, SGD, ReDial, and JDDC in terms of F1. From the results, we can observe that without task attribute fulfillment representation the performances deteriorate considerably. Thus, utilization of task attribute fulfillment representation is necessary for user satisfaction modeling.

Task Attributes	
Doctor: ▪ Type: speciality of the doctor. ▪ City: city where the doctor is located.	Importance: ★★★★★ ★★★

Dialogue Context	
U: I need a doctor. S: In what city? U: In New York. S: Do you want a general practitioner, ophthalmologist, or something else? U: I'm looking for a gynecologist. S: Borodulin Tatyana MD, a general practitioner in New York, is a good option.	
Ground Truth User Satisfaction	Dissatisfied
User Satisfaction Predicted by SG-USM	Dissatisfied
User Satisfaction Predicted by USDA	Neural

(a) Example 1

Task Attributes	
Travel: ▪ Category: category to which the attraction belongs. ▪ FreeEntry: whether entrance to attraction is free.	Importance: ★★★★★ ★★★

Dialogue Context	
U: Would you show me attractions to visit in Philadelphia? I prefer a museum, and someplace without an entry fee. S: Barnes Foundation is an art museum that you may like. U: Okay. Is it free? S: No. The ticket for an adult is $25. U: Sorry, I want a museum without an entry fee.	
Ground Truth User Satisfaction	Neural
User Satisfaction Predicted by SG-USM	Neural
User Satisfaction Predicted by USDA	Satisfied

(b) Example 2

Figure 6.3: Case study on SG-USM and USDA on SGD dataset. The yellow star represents the importance of task attributes. The texts in green are the users' preferences for the task attributes. The texts in red are the attributes of the provided solutions.

Model	MWOZ				ReDial			
	Acc	P	R	F1	Acc	P	R	F1
USDA	32.8	34.5	32.2	33.1	25.4	29.5	26.4	27.3
SG-USM(L)	40.9*	38.9*	41.3*	40.2*	30.8*	34.6*	30.7*	32.1*
SG-USM(L&U)	**43.1***	**40.9***	**43.5***	**42.8***	**32.3***	**36.4***	**32.8***	**33.4***

Table 6.3: Performance of SG-USM and the best baseline USDA on zero-shot learning ability. All the models are trained on SGD and tested on MWOZ and ReDial. Numbers in **bold** denote best results in that metric. Numbers with * indicate that the model is better than the performance of baseline with statistical significance (t-test, $p < 0.05$).

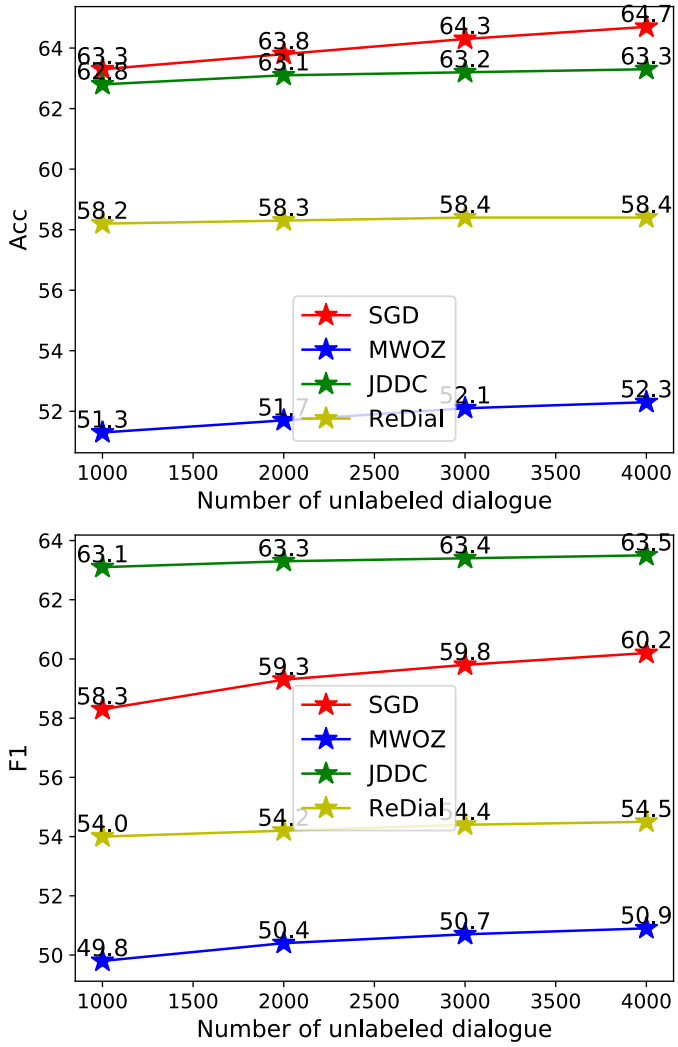


Figure 6.4: Performance of SG-USM trained with different numbers of unlabeled dialogues on MWOZ, SGD, ReDial, and JDDC datasets.

6.4.3 Discussion

Case Study

We also perform a qualitative analysis on the results of SG-USM and the best baseline USDA on the SGD dataset to delve deeper into the differences of the two models.

We first find that SG-USM can make accurate inferences about user satisfaction by explicitly modeling the fulfillment degree of task attributes. For example, in the first case in Figure 7.5, the user wants to find a gynecologist in New York. SG-USM can correctly predict the dissatisfied label by inferring that the first important task attribute "Type" is not fulfilled. In the second case, the user wants to find a museum without an entry fee. SG-USM can yield the correct neural label by inferring that the second important task attribute "FreeEntry" is not fulfilled. From our analysis, we think that SG-USM achieves better accuracy due to its ability to explicitly model how many task attributes are fulfilled and how important the fulfilled task attributes are. In contrast, the USDA does not model the fulfillment degree of task attributes, thus it cannot properly infer the overall user satisfaction.

Dealing with Unseen Task Attributes

We furhter analyze the zero-shot capabilities of SG-USM and the best baseline of USDA. The SGD, MWOZ, and ReDial datasets are English dialogue datasets that contain different task attributes. Therefore, we train models on SGD, and test models on MWOZ and ReDial to evaluate the zero-shot learning ability. Table 6.3 presents the Accuracy, Precision, Recall, and F1 of SG-USM and USDA on MWOZ and ReDial. From the results, we can observe that SG-USM performs significantly better than the baseline USDA on both datasets. This indicates that the agnostic task attribute encoder of SG-USM is effective. We argue that it can learn shared knowledge between task attributes and create more accurate semantic representations for unseen task attributes to improve performance in zero-shot learning settings.

Effect of the Unlabeled Dialogues

To analyze the effect of the unlabeled dialogues for SG-USM, we test different numbers of unlabeled dialogues during the training process of SG-USM. Figure 6.4 shows the Accuracy and F1 of SG-USM when using 1 to 4 thousand unlabeled dialogues for training on MWOZ, SGD, ReDial, and JDDC. From the results, we can see that SG-USM can achieve higher performance with more unlabeled dialogues. This indicates that SG-USM can effectively utilize unlabeled dialogues to improve the performance of user satisfaction modeling. We reason that with a larger corpus, the model can more accurately estimate the importance of task attributes.

6.5 Summary

User satisfaction modeling is an important yet challenging problem for task-oriented dialogue systems evaluation. For this purpose, we proposed to explicitly model the degree to which the user's task goals are fulfilled. Our novel method, namely SG-USM, models user satisfaction as a function of the degree to which the attributes of the user's task goals are fulfilled, taking into account the importance of the attributes. Extensive experiments show that SG-USM significantly outperforms the state-of-the-art methods in user satisfaction modeling on various benchmark datasets, i.e. MWOZ, SGD, ReDial, and JDDC. Our extensive analysis also validates the benefit of explicitly modeling the fulfillment degree of a user's task goal based on the fulfillment of its constituent task attributes. In future work, it is worth exploring the reasons of user dissatisfaction to better evaluate and improve task-oriented dialogue systems.

Chapter 7

Large Language Model Empowered End-to-End Task-Oriented Dialogue Systems

Conversational recommender systems (CRSs) aim to recommend high-quality items to users through a dialogue interface. It usually contains multiple sub-tasks, such as user preference elicitation, recommendation, explanation, and item information search. To develop effective CRSs, there are some challenges: 1) how to properly manage sub-tasks; 2) how to effectively solve different sub-tasks; and 3) how to correctly generate responses that interact with users. Recently, Large Language Models (LLMs) have exhibited an unprecedented ability to reason and generate, presenting a new opportunity to develop more powerful CRSs. In this work, we propose a new LLM-based agent for CRS, referred to as CRSAgent, to address the above challenges. For sub-task management, we leverage the reasoning ability of LLM to effectively manage sub-tasks. For sub-task solving, we collaborate LLM with expert models of different sub-tasks to achieve the enhanced performance. For response generation, we utilize the generation ability of LLM as a language interface to better interact with users. Specifically, CRSAgent divides the workflow into four stages: sub-task detection, model matching, sub-task execution, and response generation. CRSAgent also designs schema-based instruction, demonstration-based instruction, dynamic sub-task and model matching, and summary-based generation to instruct LLM to generate desired results in the workflow. Finally, to adapt LLM to conversational recommendations, we also propose to fine-tune LLM with reinforcement learning from CRSs performance feedback, referred to as RLPF. Experimental results on benchmark datasets show that CRSAgent with RLPF outperforms the existing methods.

7.1 Introduction

With the advancements in conversational intelligence in recent years, conversational recommender systems (CRSs) have attracted much more attentions [2, 9]. It usually consists of several sub-tasks, including user preference elicitation, recommendation, explanation, and item information search [214]. Figure 7.1 shows an example of CRSs where a user resorts to the agent for movie suggestions. Combining the users preference elicited through conversational interactions, the system can offer desired recommendations easily. The sys-

tem can also answer user's questions about recommendations such as why to recommend this item and more information about the item.

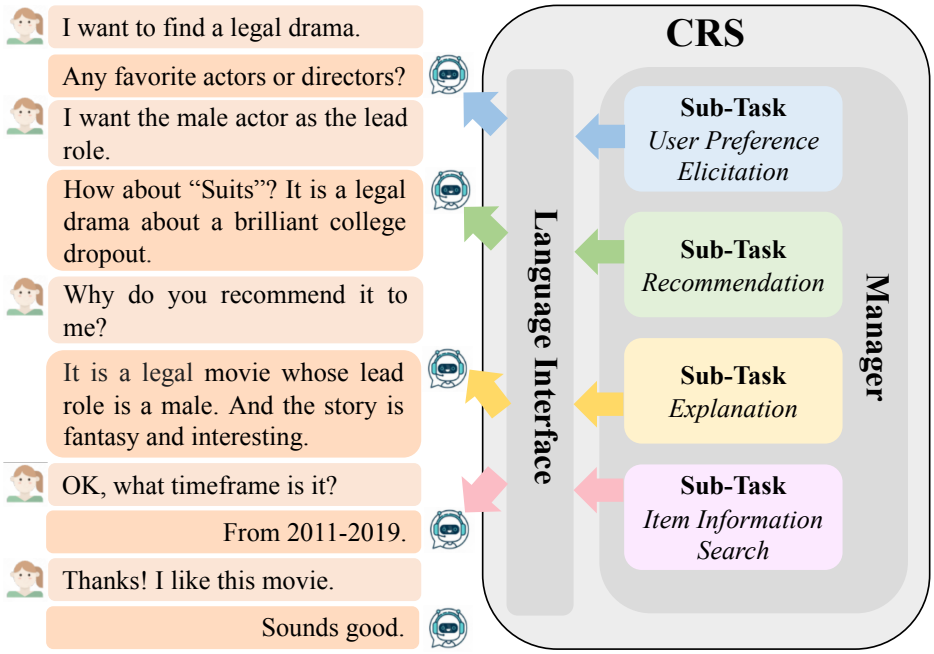


Figure 7.1: An example of a conversational recommender system (CRS). A conversational recommendation typically consists of multiple sub-tasks. A CRS must have the ability for sub-task management and sub-task resolution. It also needs to generate responses to interact with users.

To develop more effective CRSs, there are some challenges that need to be addressed: 1) First, the CRSs need a sub-task manager. Given that multiple sub-tasks are involved in conversational recommendation in a consistent dialog interface, the CRSs should effectively decide *when* to perform *which* sub-task. 2) The CRSs need to proficiently address various sub-tasks individually. The CRSs need to design and optimize for specific sub-tasks, allowing them to achieve enhanced performance. And 3) the CRSs need to have superior language generation ability. The CRSs should produce fluent, consistent, and meaningful responses to the users.

Recently, due to the remarkable understanding and generation capabilities demonstrated by the Large Language Models (LLMs) [215, 216, 217, 218, 219, 220], new possibilities emerge for creating more powerful CRSs. There are a few preliminary approaches use LLMs in CRSs, such as using the understanding ability of LLMs to rerank the results of recommendation model [146] and using the generation ability of LLMs to enhance the generated responses [221] . However, they cannot conduct sub-task management and they also cannot effectively address different sub-tasks, resulting in an inferior CRSs.

Inspired by some recent advances that use LLMs for task planning, external tools plug-in, and generation [222, 223], we propose a novel LLM-based agent for CRSs, referred to as *CRSAgent*, which 1) utilizes LLMs to effectively manage sub-tasks, 2) collaborates LLMs with several expert models of different sub-tasks to improve sub-task performance; and 3) uses LLM as a language interface to generate enhanced responses to the users. The first challenge of CRSAgent is **how to define the workflow** so that the LLM can simultaneously enhance sub-task management, facilitate collaboration with expert models,

and optimize response generation. The second challenge of CRSAgent is **how to effectively instruct LLM** to generate specific and desired outputs to improve the controllability of the system. The third challenge of CRSAgent is **how to determine the optimal format for interaction** with the LLM in the workflow to enhance the system's scheduling and collaboration capabilities. The fourth challenge of CRSAgent is **how to adapt LLM to conversational recommendation data** to improve the overall performance of the system.

Specifically, the workflow of CRSAgent is divided into four stages: *sub-task detection*, *model matching*, *sub-task execution*, and *response generation*. Besides, CRSAgent designs different mechanisms to effectively instruct LLM to generate desired results. 1) It uses *schema-based instruction* to instruct LLM to understand the characteristics of different sub-tasks. 2) It uses *demonstration-based instruction* to instruct LLM to understand the criteria for sub-task detection. 3) It uses a *dynamic sub-task and model matching mechanism* to instruct LLM to select the suitable expert model from the dynamic candidate model set. 4) It uses the *summary-based generation* to instruct LLM to incorporate all the information of the previous stages to generate responses. Additionally, the instruction mechanisms are augmented by designed prompts which are subsequently utilized as the interaction format for CRSAgent to function effectively. Finally, we propose a new method, referred to as *RLPF*, which utilizes reinforcement learning from CRSs performance feedback to fine-tune LLMs to adapt to conversational recommendation data to achieve better performance. RLPF uses *recommendation feedback* and *conversation feedback* as reward signals and uses the REINFORCE [224] method to guide the learning direction. In order to lead to more stable updates and learning, we also employ a baseline function to reduce the variance of the estimated reward.

We conduct experiments on two benchmark datasets for conversational recommendation, which are GoRecDial and TG-ReDial datasets. The experimental results show that CRSAgent substantially outperforms existing methods on recommendation and also provides a more satisfying natural language interaction. The extensive analysis also reveals the significance of sub-task management, the impact of cooperating with expert models of sub-tasks, the efficacy of utilizing reinforcement learning from CRSs performance feedback, and the effectiveness of instruction mechanisms for LLM.

To sum up, our contributions are as follows:

- We propose a new conversational recommendation agent, referred to as CRSAgent, which effectively utilizes LLM to address challenges in CRSs, including sub-task management, proficient sub-tasks handling, and advanced language generation.
- CRSAgent introduces a novel workflow for conversational recommendation agents, incorporating diverse mechanisms and carefully crafted prompts to instruct and interact efficiently with LLMs in the context.
- CRSAgent is refined by reinforcement learning from CRSs performance feedback (RLPF) to get an enhanced performance.
- The experiment results of CRSAgent achieve the best performances against baselines on recommendation and have better natural language interaction. Extensive experimental analysis also helps to better understand the advantages of our method.

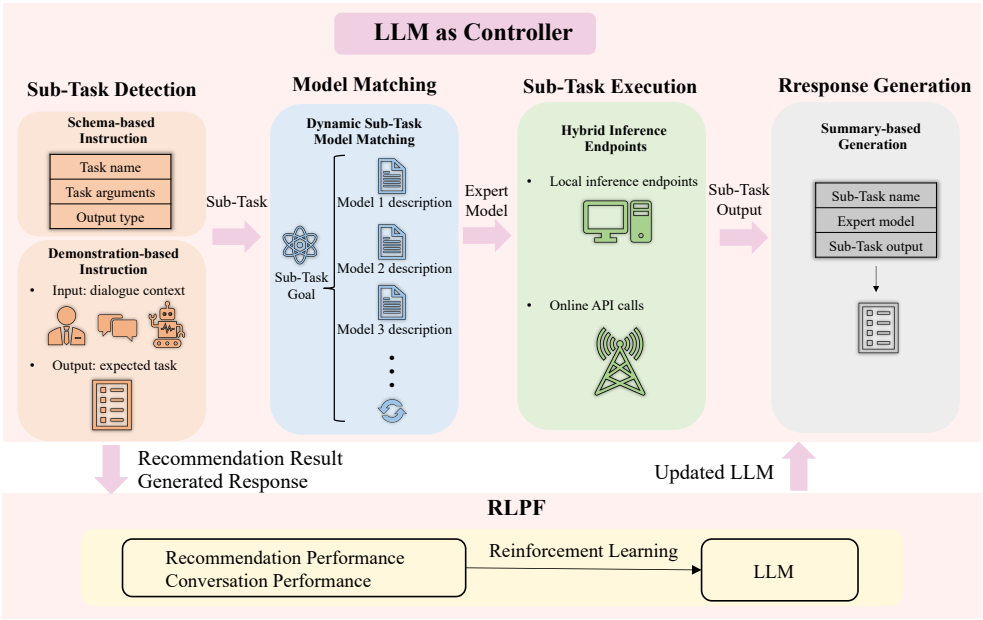


Figure 7.2: The framework of CRSAgent. It consists of an LLM as the controller and numerous expert models as collaborative executors. The LLM rst detects the sub-task. Then it selects a suitable expert model for the sub-task. After that, the sub-task is assigned to the expert model to get the results. Finally, LLM collects the results from the expert model and generates responses to the user. The Reinforcement Learning from CRSs Performance Feedback (RLPF) is used to adapt the LLM to CRSs.

7.2 Framework

CRSAgent is a collaborative conversational recommender system that consists of an LLM as the manager and numerous expert models as collaborative executors. The workflow of CRSAgent consists of four stages: sub-task detection, model matching, sub-task execution, and response generation, as shown in Figure 7.2. Specifically,

1. An LLM first analyzes the dialogue context to detect which sub-task should be executed for the current dialogue turn based on its knowledge;
2. The LLM distributes the detected sub-task to a suitable expert model according to the model descriptions;
3. The expert model executes the assigned sub-task on the inference endpoint and returns the execution information and inference results to LLM;
4. Finally, the LLM summarizes the execution information and inference results, and generates the response to the user.

7.2.1 Sub-Task Detection

In the first stage of CRSAgent, the LLM takes the dialogue context as input and detects which sub-task should be executed for the current dialogue turn. Complex conversational recommender systems often involve several sub-tasks and different sub-tasks requiring different sub-task arguments. Schema can be

Table 7.1: Sub-tasks schema in CRSAgent.

Sub-Task Name	Sub-Task Arguments	Output type
User Preference Elicitation	{category, dialogue context}	Text
Recommendation	{category, dialogue context}	Item
Explanation	{item name, dialogue context}	Text
Item Detail Search	{item name, item attribute}	Text

used to provide a uniform template for different sub-tasks. And demonstrations can also instruct LLM to get better performance. Therefore, to unify different tasks and improve the LLM's understanding of the criteria for sub-task detection, CRSAgent employs schema-based instruction and demonstration-based instruction in its prompt design. The prompt is shown in Table 7.2. We introduce the details in the following paragraphs.

Schema-based Instruction:

Schema can be used to provide a uniform template for different sub-tasks in the conversational recommender system. CRSAgent designs three slots in the schema for sub-task detection, which are the sub-task name, sub-task arguments, and output type:

- *Sub-task name*: It provides a unique identifier for sub-task detection, which is used for references to related expert models and their predicted results. The list of currently supported subtasks of CRSAgent is shown in Table 7.1.

- *Sub-task arguments*: It contains the list of required arguments for sub-task execution. They are resolved from either the dialogue context or the predicted results of the sub-tasks executed in the previous dialogue turns. The corresponding arguments for different sub-tasks are shown in Table 7.1.

- *Output type*: It indicates the type of the outputs of the sub-task. The corresponding types of outputs for different sub-tasks are shown in Table 7.1.

Demonstration-based Instruction:

By injecting several demonstrations into the prompts, CRSAgent can allow the large language model to better understand the criteria for sub-task detection. Each demonstration is a set of inputs and outputs on sub-task detection, which are the dialogue context and the expected sub-tasks to be executed. The sub-task in the demonstrations is represented by schema, which is inferred by the LLM. Demonstrations can effectively aid CRSAgent in understanding the criteria for sub-task detection and the semantic meaning of the slots in the schema.

Benefiting from instruction tuning [225] and reinforcement learning from human feedback [216], the large language model has the ability to follow instructions. CRSAgent provides these schema-based instructions and demonstration-based instructions to the large language model as high-level instructions for analyzing the dialogue context and detecting sub-tasks accordingly.

Table 7.2: The details of the prompt design in CRSAgent. There are some injectable slots in the prompts, such as Sub-Task List, Sub-Task Schema, Demonstrations, Dialogue Context, Sub-Task Goal, Expert Models, and Sub-Task Output. These slots are uniformly replaced with the corresponding text before being fed into the LLM.

Stage	**Prompt**
Sub-Task Detection	The AI assistant should analyze the dialogue context to detect which sub-task should be selected. The sub-tasks include: *{{ Sub-Task List }}*. The selected sub-task need to represent by its corresponding sub-task schema. The sub-task schema is *{{ Sub-Task Schema }}*. There are some cases for your reference: *{{ Demonstrations }}*. The dialogue context is *{{ Dialogue Context }}*. From this dialogue context, which sub-task should be selected? The sub-task MUST be selected from the above list and represented by the schema.
Model Matching	The AI assistant should select the most appropriate expert model to process the sub-task based on the sub-task goal and expert model description. The sub-task goal is *{{ Sub-Task Goal }}*. The list of expert models is *{{ [ID, Description] }}*. Please select one expert model. The expert model MUST be selected from the above list and represented by the ID.
Response Generation	With the dialogue context and the sub-task results, the AI assistant needs to generate a response to the user. The dialogue context is *{{ Dialogue Context }}*. The sub-task results can be formed as: Sub-Task Name: *{{ Sub-Task Name }}*, Expert Model: *{{ ID, Description }}*, and Sub-Task Output: *{{ Output }}*. Please generate a response to answer the users request.

7.2.2 Model Matching

After detecting the sub-task to be executed, CRSAgent next needs to match the sub-task and the expert model, that is to say, select an appropriate model from the candidate expert model set for the sub-task. To make CRSAgent have the scalability to the candidate expert model set, we propose a dynamic sub-task and model matching mechanism, which can select a model from the dynamic candidate expert model set. This practice makes CRSAgent more open and flexible. The prompt is shown in Table 7.2. We introduce the details in the following paragraphs.

Dynamic Sub-Task and Model Matching:

To make CRSAgent have incremental expert model access ability, we approach the matching of sub-tasks and models as matching problem between sub-task goal and model description, where sub-tasks and expert models are presented using text in the prompt. When new expert models are added to the system, we just simply add the new expert model descriptions to the prompt and analyze the relevance between the new model description and the sub-task goal. Model descriptions are often provided by the developers, which contain information on functions, architecture, supported languages, domains, licensing, and more. This information can help LLM to understand the functionality of the expert models. Therefore, presenting sub-task using its goal and presenting expert models using model descriptions in the prompt can support CRSAgent selecting the suitable model for the sub-task from a set of candidate expert models.

7.2.3 Sub-Task Execution

Once a sub-task is assigned to a specific expert model, the next step is to execute the sub-task, i.e., perform the expert model inference process. By taking the sub-task arguments at the sub-task detection stage as inputs, the expert model computes the sub-task outputs and then sends them back to the LLM. Besides, at this stage, to address the expert model availability problem and data security problem, the expert models should be run on hybrid inference endpoints, which are online API calls and local inference endpoints.

Hybrid Inference Endpoints:

An ideal scenario is that we only use inference endpoints by online API calls, which can help to reduce computing resources and save storage space. However, in some cases, we have to deploy local inference endpoints, for example when inference endpoints for certain expert models do not exist, the inference process is time-consuming, or network access is limited. To solve the above issues and keep the system stable and efficient, CRSAgent pulls and runs common or time-consuming models locally and the other models online. Besides, due to the security of the data and user privacy, CRSAgent requests local inference endpoints with higher priority than online API calls. Only if the matched expert model is not deployed locally, CRSAgent will run the expert model by online API calls.

7.2.4 Response Generation

After sub-task execution is completed, CRSAgent enters the response generation stage. To better instruct LLM to generate the response to the user,

CRSAgent needs to utilize dialogue context and all the information of the previous stages, including sub-task detection, model matching, and sub-task execution as inputs to the LLM. Therefore, we propose a summary-based generation practice. The prompt is shown in Table 7.2. We introduce the details in the following paragraphs.

Summary-based Generation:

To incorporate all the information of the previous stages, CRSAgent takes a summary as inputs of LLM. To make the summary concise, we represent the summary in a structured format, which includes three attributes as follows,

- *Sub-task name*: The detected sub-task for current dialogue turn.
- *Expert model*: The description of the selected expert model for the detected sub-task.
- *Sub-Task output*: The inference results of the expert model.

CRSAgent allows the LLM to generate the final responses using the summary and dialogue context as input. It can effectively incorporate all the information from the previous stages and the dialogue context, helping to provide more instructions to the LLM.

7.3 Reinforcement Learning from CRSs Performance Feedback

While learning only from prompts is a powerful method for instructing LLM, it is not sufficient to solve real-world recommendation problems that require a deeper understanding of dialogue context and recommendation environment. One potential method to improve the understanding capabilities of LLM is to use reinforcement learning techniques to fine-tune LLM for conversational recommendation problems. In this work, we propose Reinforcement Learning from CRSs Performance Feedback (RLPF), which uses recommendation performance and response generation performance to guide LLM learning, resulting in improved overall performance for CRSs.

In the setup of RLPF, the environment is the proposed CRSAgent platform and the agent is the large language model L parameterized with Θ. The solutions S generated by the LLM can be seen as actions that are used to solve the conversational recommendation problem. We can use the performance on that dataset as the reward signal and use reinforcement learning to fine-tune the LLM. More concretely, to find the optimal solution, we require the LLM to maximize its expected reward $\mathcal{R}$ on the training set T_{train}, represented by $J(\Theta)$:

$$J(\Theta) = \mathbb{E}_{S \sim L(T_{train}|\Theta)}[\mathcal{R}]. \tag{7.3.1}$$

The reward $\mathcal{R}$ is composed of the recommendation evaluation metric HIT [226] and the response generation evaluation metric BLEU [100] with balance parameter λ. Here is a preliminary attempt to demonstrate the effectiveness of RLPF learning framework. In future work, we can also explore more other evaluation metrics.

$$\mathcal{R} = \lambda \cdot \text{HIT} + (1 - \lambda) \cdot \text{BLEU}. \tag{7.3.2}$$

Since the reward signal $\mathcal{R}$ is non-differentiable, we need to use a policy gradient method to iteratively update Θ. In this work, we use the REINFORCE [224] method to update the parameters as follows,

$$\nabla_\Theta J(\Theta) = \mathbb{E}_{p(S|\Theta)}[\nabla_\Theta \log P(S|\Theta) \cdot \mathcal{R}]. \tag{7.3.3}$$

An empirical approximation of the above quantity as follows,

$$\nabla_\Theta J(\Theta) \approx \frac{1}{|T_{train}|} \sum_{t \in T_{train}} \nabla_\Theta \log P(S|\Theta) \cdot \mathcal{R}. \tag{7.3.4}$$

The above update is an unbiased estimate for the gradient but has a very high variance. In order to reduce the variance of this estimate, following previous work [227], we employ a baseline function b, which is a moving average of the previous reward signals:

$$\nabla_\Theta J(\Theta) \approx \frac{1}{|T_{train}|} \sum_{t \in T_{train}} \nabla_\Theta \log P(S|\Theta) \cdot (\mathcal{R} - b(S)). \tag{7.3.5}$$

The RLPF approach can effectively refine the large language models to recommendation and response generation, leading to a significant improvement for CRSs.

7.4 Experiments

7.4.1 Dataset

We conduct the experiments using the benchmark datasets on conversational recommendation, namely GoRecDial [228] and TG-ReDial [140]. The introduction of the datasets is as follows.

- **GoRecDial**: It is a conversational recommendation dataset released by Kang et al. [228]. This dataset was constructed using ParlAI [229] to interface with Amazon Mechanical Turk (AMT)[1]. To reflect the movie preferences of real users, this dataset built the pool of recommendation movies using the MovieLens dataset[2], comprising 27M ratings applied to 58K movies by 280K real users. To obtain the movie information, they obtained the descriptive text for each movie from Wikipedia[3] and extracted entity-level features (e.g., directors, actors, year) using the MovieWiki dataset [230]. The statistics of GoRecDial are presented in Table 7.3.

- **TG-ReDial**: It is a conversational recommendation dataset released by Zhou et al. [140]. The conversation was created in a semi-automatic way by involving reasonable and controllable human annotation efforts [231]. The movie watching records was collected from real users on Douban[4] website. The dataset contains 1,482 users and 202.7 watching records for each user on average. The movie information was extracted from movie tags on Douban (e.g. genre, director, and starring). The statistics of TG-ReDial are presented in Table 7.3.

[1] https://www.mturk.com/
[2] https://grouplens.org/datasets/movielens/
[3] https://dumps.wikimedia.org/
[4] https://www.douban.com/

Table 7.3: Statistics of GoRecDial and TG-ReDial in the experiments.

Characteristics	GoRecDial	TG-ReDial
#Dialogues	9,125	10,000
#Utterances	170,904	129,392
Avg turn per dialogue	9.3	6.5
Avg token per utterance	8.4	19.0
#Item	5,300	33,834

7.4.2 Baselines

Following [140, 141, 139], we evaluate the superiority of our method by considering the following representative baselines:

- **BERT** [22]: It is a bidirectional language model pre-trained via the masked language modeling task on a large-scale general corpus. We utilize the representation of the [CLS] token for recommendation. We reproduce the baseline using their open-source code.

- **GPT-2** [44]: It is an autoregressive language model pre-trained via the language modeling task on large-scale general corpora. We take the generated text of the language model for response and use the representation of the last token for recommendation. We reproduce the baseline using their open-source code.

- **ReDial** [137]: It includes a conversation module based on sequence-to-sequence learning and a recommendation module based on a denoising auto-encoder. We reproduce the baseline using their open-source code.

- **KBRD** [139]: It utilizes a knowledge graph to enhance the semantics of contextual items or entities for recommendation. The dialog generation module is based on the Transformer [182] architecture. We reproduce the baseline using their open-source code.

- **KGSF** [141]: It incorporates both word-oriented and entity-oriented knowledge graphs to enhance the performance of conversational recommendation. We reproduce the baseline using their open-source code.

- **TG-ReDial** [140]: It adopts BERT to encode the historical utterances to model the dialogue topic, and leverages graph neural networks for topic-guided item recommendation and response generation. We reproduce the baseline using their open-source code.

7.4.3 Evaluation Measures

Following existing work [140, 141, 139], we adopt different metrics to evaluate the performance of recommendation and conversation.

For the recommendation, we develop ranking-based metrics for measuring the ranking performance of the generated recommendation lists, which include

$HIT@k$ [226], $MRR@k$ [232], and $NDCG@k$ [233] (k = 1, 10, and 50). We give the precise definition of these metrics as follows.

$$HIT@k = \frac{|\{j \in R_k | r_j = 1\}|}{|k|}, \tag{7.4.1}$$

where R_k is the top k ranking list, and r_j is the relevance score of item j.

$$MRR@k = \frac{1}{N} \sum_{j \in R_k} \frac{r_j}{rank_j}, \tag{7.4.2}$$

where N is the number of test users, r_j is the relevance score of item j, and $rank_j$ is the position of an item j in the top k ranking list R_k.

$$DCG@k = \sum_{j=1}^{k} \frac{2^{r_j} - 1}{log_2(1+j)}, \tag{7.4.3}$$

$$NDCG@k = \frac{DCG@k}{IDCG@k}, \tag{7.4.4}$$

where r_j is the relevance score of item j, and $IDCG$ is the DCG score for the most ideal ranking.

For the conversation, we use both relevance-based and diversity-based evaluation metrics to measure the performance of generated responses. The relevance-based metrics include BLEU [100] which measure the similarity between ground truth and generated responses from the perspective of probability. The diversity-based metrics are Distinct [234], measuring the number of distinct in the generated responses.

A paired t-test is conducted to determine whether there is a significant difference between our method and the best baseline. We compare our method with TG-ReDial. The significance level (p-value) is set at 0.05.

7.4.4 Implementation Details

For fair comparisons, we implement all the baselines and CRSAgent by the open-source toolkit CRSLab[5] [235]. The hyper-parameter settings of the baselines follow the default settings on CRSLab, which reaches the best performances. The data preprocessing is also consistent with that of CRSLab, ensuring a fair and equitable comparison. The expert models we use in CRSAgent are the state-of-the-art methods for each task. Specifically, for the user preference elicitation task, we use KBRD [139] and KGSF [141] methods; for the recommendation task, we use TG-ReDial [140] method; for the explanation task, we use KBRD [139] and KGSF [141] methods; for the item detail search task, because the results are searched directly from the database, we do not use any expert models.

In addition, we employ our system using two open-source large language models, which are Flan-T5-*Large* [**chung2022scaling**] and LLaMA-*7b* [220].

- **Flan-T5**: It is a series of language models developed by Google. It is fine-tuned using instruction-tuning, which allows them to have good performance on a variety of tasks. In our work, we use Flan-T5-*Large*, which has 770 million parameters.

[5]https://github.com/RUCAIBox/CRSLab

- **LLaMA**: It is a lightweight, open-source language model developed by researchers at Meta. It is designed to be efficient and performant. In our work, we use LLaMA-*7b*, which has 7-billion parameters.

Moreover, we use the Adam [178] optimizer to update the parameters. The learning rate and the weight decay rate are set to be 5e-5 and 0.01, respectively. The batch size is 1. The max source sequence length and the max target sequence length are 512 and 100, respectively. We train the model on eight NVIDIA GeForce RTX 2080-12GB GPUs and the training time is around 36h.

7.5 Experimental Results and Discussion

7.5.1 Overall Performance

Tables 7.4, 7.5, 7.6, and 7.7 show the performance of CRSAgent as well as the baselines of recommendation and conversation on GoRecDial and TG-ReDial datasets respectively. It is shown that CRSAgent achieves state-of-the-art performance in the recommendation which is the most important performance for CRSs. All improvements observed compared to the baselines are statistically significant according to two sided paired t-test ($p < 0.05$). And CRSAgent can also provide a more satisfying language interface compared to the state-of-the-arts. The performance of CRSAgent on relevance-based conversation evaluation metric is similar to the baselines. Most notably, CRSAgent can significantly improve the performance of the diversity-based evaluation metric on the conversation. It indicates that CRSAgent can keep the consistency with the ground truth dialogue and also generate more diversity and informative responses to the user, resulting in the improved interaction between the user and the system. We conjecture that the overall good performance of CRSAgent is due to it containing the LLM, which has exhibited exceptional ability in task planning, tool interaction, language understanding, and language generation. The following analysis provides a better understanding of our models strengths.

7.5.2 Ablation Study

We conduct an ablation study on CRSAgent to quantify the effects of three factors: the usage of sub-task management, the cooperation with the expert models, and the usage of the reinforcement learning from CRSs performance feedback. The results indicate that all above factors of CRSAgent are indispensable for conversational recommendation.

Effect of the usage of sub-task management

To investigate the effectiveness of sub-task management, we compare CRSAgent with CRSAgent-w/o M which eliminates the sub-task management in the system. CRSAgent-w/o M only interacts with one expert model TG-ReDial and generates responses based on the outputs of TG-ReDial, no matter which sub-task the current dialogue should solve. Figure 7.3 and 7.4 show the results of CRSAgent-w/o M on GoRecDial and TG-ReDial datasets in terms of HIT@10, MRR@10, NDCG@10, BLEU-1, and Distinct-1. From the results, we can observe that without sub-task management, the performances

Table 7.4: Performance of CRSAgent and baselines for recommendation on GoRecDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$).

Model	HIT@1	HIT@10	HIT@50	MRR@1	MRR@10	MRR@50	NDCG@1	NDCG@10	NDCG@50
BERT	0.0458	0.2212	0.4827	0.0458	0.0905	0.1024	0.0458	0.1212	0.1780
GPT-2	0.0111	0.1095	0.2927	0.0111	0.0305	0.0376	0.0111	0.0484	0.0866
ReDial	0.0483	0.2185	0.4599	0.0483	0.0948	0.1123	0.0483	0.1295	0.1805
KBRD	0.0581	0.2268	0.4626	0.0581	0.1037	0.1150	0.0581	0.1326	0.1848
KGSF	0.0592	0.2603	0.5374	0.0592	0.1074	0.1206	0.0592	0.1429	0.2042
TG-ReDial	0.0519	0.2853	0.5496	0.0519	0.1138	0.1267	0.0519	0.1539	0.2129
CRSAgent (Flan-T5)	0.0612	0.3022	0.5714	0.0612	0.1253	0.1501	0.0612	0.1723	0.2286
CRSAgent (LLaMA)	**0.0635**	**0.3031**	**0.5718**	**0.0635**	**0.1296**	**0.1518**	**0.0635**	**0.1783**	**0.2290**

Table 7.5: Performance of CRSAgent and baselines for recommendation on TG-ReDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline (two-sided paired t-test, $p < 0.05$).

Model	HIT@1	HIT@10	HIT@50	MRR@1	MRR@10	MRR@50	NDCG@1	NDCG@10	NDCG@50
BERT	0.0072	0.0049	0.0281	0.0072	0.0106	0.0124	0.0049	0.0147	0.0239
GPT-2	0.0021	0.0192	0.0421	0.0021	0.0051	0.0082	0.0021	0.0102	0.0187
ReDial	0.0028	0.0249	0.0533	0.0028	0.0073	0.0104	0.0028	0.0112	0.0203
KBRD	0.0040	0.0254	0.0588	0.0040	0.0089	0.0103	0.0040	0.0127	0.0198
KGSF	0.0053	0.0285	0.0771	0.0053	0.0114	0.0135	0.0053	0.0154	0.0259
TG-ReDial	0.0079	0.0251	0.0524	0.0079	0.0122	0.0134	0.0079	0.0152	0.0211
CRSAgent (Flan-T5)	0.0084	0.0302	**0.0792**	0.0084	0.0128	0.0138	0.0084	0.0159	0.0261
CRSAgent (LLaMA)	**0.0086**	**0.0308**	0.0791	**0.0086**	**0.0130**	**0.0139**	**0.0086**	**0.0162**	**0.0263**

Table 7.6: Performance of CRSAgent and baselines for conversation on GoRecDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline on Distinct (two-sided paired t-test, $p < 0.05$).

Model	BLEU-1	BLEU-2	Distinct-1	Distinct-2
GPT-2	0.0120	0.0020	0.0219	0.1542
ReDial	0.0421	0.0035	0.0044	0.0124
KBRD	**0.0782**	**0.0068**	0.0081	0.0235
KGSF	0.0467	0.0037	0.0078	0.0124
TG-ReDial	0.0226	0.0046	0.0120	0.0960
CRSAgent (Flan-T5)	0.0528	0.0050	**0.1659**	**0.2944**
CRSAgent (LLaMA)	0.0601	0.0057	0.1590	0.2903

Table 7.7: Performance of CRSAgent and baselines for conversation on TG-ReDial. Numbers in **bold** denote the best results in that metric. CRSAgent significantly improves over the best baseline on Distinct (two-sided paired t-test, $p < 0.05$).

Model	BLEU-1	BLEU-2	Distinct-1	Distinct-2
GPT-2	0.0858	0.0119	2.3500	4.6200
ReDial	0.0570	0.0044	0.0041	0.0070
KBRD	0.2670	0.0458	0.4690	1.5000
KGSF	**0.3830**	**0.1150**	0.3400	0.9100
TG-ReDial	0.1250	0.0204	0.8810	1.7500
CRSAgent (Flan-T5)	0.3011	0.1021	**2.4231**	**4.8332**
CRSAgent (LLaMA)	0.3123	0.1088	2.4128	4.8023

of recommendation and conversation deteriorate considerably. It indicates that sub-task management, which effectively detects when to do which sub-task, can better understand the process of conversation and improve the overall performance of CRSs.

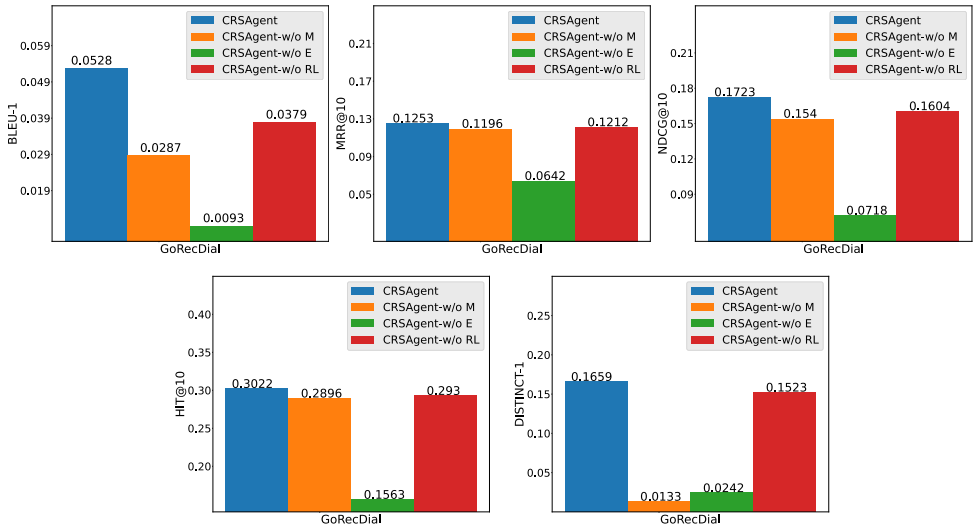


Figure 7.3: Ablation study of CRSAgent with respect to the sub-task management, expert models, and RLPF on GoRecDial dataset.

Effect of Cooperation with Expert Models

To validate the effectiveness of incorporating with the expert models, we employ each sub-task with the LLM instead of using expert models. For user preference elicitation, explanation, and item information search, we directly take the dialogue context as input of LLM and then use the responses generated by the LLM as results. For recommendation, due to the candidate item set being huge, it is hard to take all of it as input of LLM. Therefore, we first calculate the concise similarity of the item and dialogue by BERT [22] and select the top 50 relevant items based on the similarity score. Then, LLM takes the small set of candidates, similarity scores, and dialogue context as input to predict the recommendation. Figure 7.3 and 7.4 show the results of CRSAgent-w/o E on

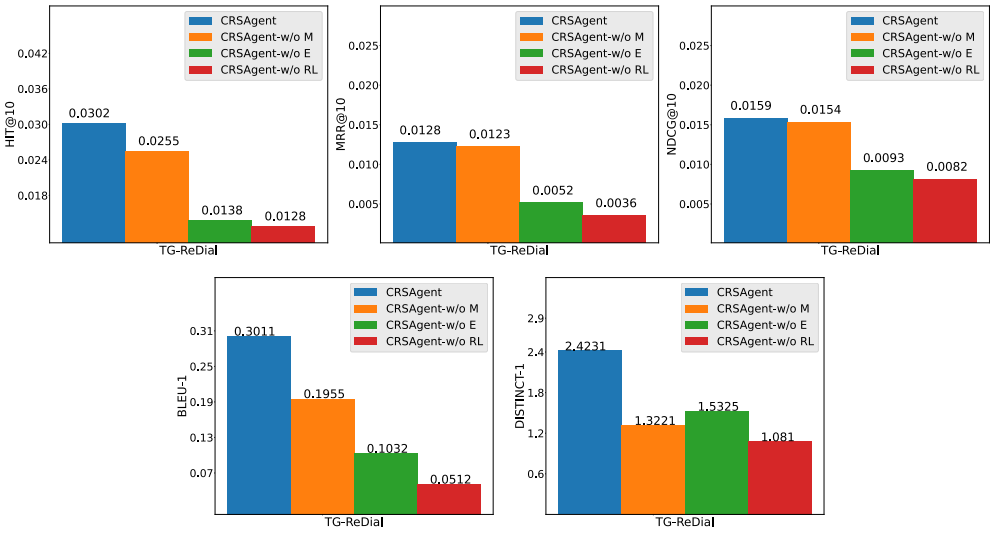


Figure 7.4: Ablation study of CRSAgent with respect to the sub-task management, expert models, and RLPF on TG-ReDial dataset.

GoRecDial and TG-ReDial datasets. We can see that the performance of the CRSAgent-w/o E in terms of HIT@10, MRR@10, NDCG@10, BLEU-1, and Distinct-1 decreases significantly compared with CRSAgent. It indicates that cooperation with expert models can help CRSAgent to have a bridge between LLM and the task-specic models, allowing for effective knowledge transfer and improved performance. In addition, the plug-in expert model mechanism also supports dynamically adding new expert models in the system, which enhances the scalability and flexibility of CRSs.

Effect of RLPF

To analyze the effectiveness of reinforcement learning from CRSs performance feedback (RLPF), we compare the performance of CRSAgent with CRSAgent-w/o RL which eliminates the reinforcement learning mechanism in the system. Figure 7.3 and 7.4 show the results of CRSAgent-w/o RL and CRSAgent on GoRecDial and TG-ReDial datasets in terms of HIT@10, MRR@10, NDCG@10, BLEU-1, and Distinct-1. One can observe that without RLPF, the performances deteriorate considerably. The RLPF mechanism effectively renes the LLMs recommendation ability and response generation strategy, resulting in an enhanced and more adaptive CRS. We think that it is due to only relying on the input text for learning is insufcient for LLM when solving the conversational recommendation problem. Performance feedback can offer valuable supplementary information that steers the learning trajectory of LLMs, enabling them to furnish more precise recommendations and generate more fitting responses. In addition, the absence of RLPF results in a more pronounced decline in the performance of TG-ReDial compared to ReDial. We speculate that this phenomenon arises from the fact that conversations in TG-Redial are structured using predefined topic threads. Consequently, CRSs' ability to adapt to specific topics becomes crucial for achieving improved performance. This phenomenon further demonstrates the superior adaptability of RLPF.

Table 7.8: Effectiveness of the mechanisms to instruct LLM in CRSAgent. Numbers in bold denote the best results.

Model	HIT@10	MRR@10	NDCG@10	BLEU-1	Distinct-1
CRSAgent (Flan-T5)	**0.3022**	**0.1253**	**0.1723**	**0.0528**	**0.1659**
-w/o SI	0.2963	0.1228	0.1674	0.0412	0.1588
-w/o DI	0.3003	0.1236	0.1692	0.0474	0.1602
-w/o SG	0.2992	0.1235	0.1691	0.0460	0.1591

7.5.3 Mechanisms to Instruct LLM

The mechanisms to instruct LLM include schema-based instruction, demonstration-based instruction, dynamic sub-task and model matching, and summary-based generation. Due to the limited number of expert models we can access, we do not analyze the dynamic sub-task and model matching mechanism. Our analysis mainly focuses on the remaining three mechanisms. We compare CRSAgent with CRSAgent-w/o SI which removes the task schema description in the prompt, CRSAgent-w/o DI which eliminates the demonstrations in the prompt, and CRSAgent-w/o SG which directly generates responses from the execution results of the expert models instead of the structured summary text. Table 7.8 shows the results of these methods on GoRecDial in terms of HIT@10, MRR@10, NDCG@10, BLEU-1, and Distinct-1. From the results, we can observe that without any mechanism in the prompts, the performances of recommendation and conversation deteriorate considerably. We conjecture that it is due to:

1. the schema-based instruction can unify different tasks in the same format to better guide LLM to detect different tasks and also facilitate knowledge sharing among tasks;

2. the demonstration-based instruction can condition on the in-distribution conversational recommendation demonstration to make the conversational recommendation task closer to an LLM, and demonstrations can also keep the format of the input-label pairs;

3. the summary-based generation can integrate all the information from the previous stages and provide global instruction to LLM for improved response generation.

7.5.4 Case Study

We make the qualitative analysis on the results of CRSAgent and the baselines TG-ReDial on GoRecDial datasets. We find that CRSAgent can provide accurate recommendations and generate more informative responses. For example in Figure 7.5, CRSAgent can coordinate the results of expert models in the previous turns on user preference elicitation task, which are "Robert Zemeckis / Scorsese" and "Dramas" to recommend "Schindler's list" to the user. CRSAgent can also leverage the language understanding and generation ability of LLM to generate more informative and interpretative responses, such as " It is a fantastic drama", "Its story is well-crafted, and the ending is unexpected". In contrast, TG-ReDial makes a wrong recommendation.

Dialogue	U: I love a lot of different kinds of movies. Dramas and comedy are good. S: Do you have a particular director that you enjoy? U: Robert Zemeckis is good as well as Scorsese. S: Do you tend to like war dramas? U: Dramas.
Response	**Ground Truth**: Recommend Schindler's list. It seems similar to Scorsese in terms of director styles. And you like dramas.
	TG-ReDial: Recommend Little Miss Sunshine. The director is similar to the Scorsese.
	CRSAgent: Recommend Schindler's list. It's directorial style appears to be most similar to Scorsese's. And it is a fantastic drama. It's story is well-crafted, and the ending is unexpected.

Figure 7.5: Case study of CRSAgent

One potential explanation is that TG-ReDial lacks the ability to manage sub-tasks and offer solutions tailored to specific sub-tasks. Therefore, it can not accurately extract user preferences and provide accurate recommendations. Besides, the conversation ability of TG-ReDial is inferior to the LLM, which cannot generate more informative and natural responses.

7.6 Summary

We have proposed a new LLM-based agent for conversational recommender systems, referred to as CRSAgent. It uses LLM to better manage sub-tasks, effectively cooperate with expert models, and generate improved responses. The workflow of CRSAgent includes sub-task detection, model matching, sub-task execution, and response generation. At each stage, instruction learning and context learning are used to instruct LLM to perform accuratly. We also use reinforcement learning from CRSs performance feedback to refine LLM to provide more accurate recommendation and generate more suitable responses. CRSAgent is a controllable and adaptable system for conversational recommendations. Experimental results show that CRSAgent significantly outperforms the state-of-the-art methods in CRSs on the benchmark datasets of GoRecDial and TG-ReDial. Finally, we also note that the recent rapid development of LLMs has brought a huge impact on academia and industry. We also expect the design of our model can inspire the whole community and pave a new way for LLMs towards the recommendation.

Chapter 8

Future Work

As task-oriented dialogue systems (TDS) become increasingly integrated into various domains such as customer service, healthcare, finance, and education, their evolution is guided by the need to address several fundamental challenges. These challenges include trustworthiness, tool usage, reasoning, robustness, and efficiency. Below, we expand on each of these areas with a detailed description of potential directions for future research and development:

1. Trustworthiness

Trust is one of the most significant factors affecting the adoption and long-term use of task-oriented dialogue systems. Users need to trust that the system will provide reliable, accurate, and relevant information. Future research can focus on the following areas to build more trustworthy systems: 1) Fact-Checking and Verifiability: Designing systems that can verify responses against trusted databases or sources in real-time to ensure accuracy. 2) Bias Mitigation: Reducing biases in responses by improving training datasets and using techniques like adversarial debiasing and explainable AI. 3) Transparency: Providing users with explanations for the systems actions or recommendations to build trust and understanding. 4) Data Privacy: Ensuring that user data is securely handled and protected, especially in applications requiring personal information, by incorporating federated learning and robust encryption methods.

2. Tool Usage

Task-oriented dialogue systems are often required to interact with external tools and services to complete tasks. These tools could range from simple APIs to complex multi-modal systems (e.g., booking flights, making purchases, querying databases, or controlling smart home devices). The following areas need attention in future work: 1) Dynamic Tool Integration: Developing frameworks that allow systems to connect with APIs, plugins, or external databases dynamically during interactions. 2) Multi-Modal Interactions: Enabling systems to interact with tools that process various data types, such as images, audio, and structured data, to provide richer responses. 3) Error Recovery: Building mechanisms to detect and recover from errors in tool interactions, ensuring seamless user experiences. 4) Personalization: Leveraging tools to customize interactions based on user preferences, history, or real-time contextual data.

3. Reasoning

Effective reasoning is a fundamental capability for task-oriented dialogue systems, particularly when handling complex, multi-step tasks. Current sys-

tems often struggle to perform deep reasoning over extended dialogues. Future work in reasoning could address the following aspects: 1) Multi-Step Reasoning: Equipping systems with the ability to solve multi-step problems by maintaining context and breaking tasks into manageable sub-tasks. 2) Hybrid Reasoning Models: Combining neural network-based reasoning with symbolic reasoning approaches to enable better handling of logic-based queries and complex workflows. 3) Contextual Understanding: Enhancing the ability to infer user intentions from implicit cues and long-term conversation history. 4) Knowledge Augmentation: Integrating external knowledge graphs and dynamic reasoning modules to provide more comprehensive and accurate responses.

4. Robustness

Task-oriented dialogue systems must be robust enough to handle a wide variety of inputs, including noisy, ambiguous, or incomplete queries. Building robustness into these systems will ensure they can operate effectively in real-world environments. Future research could address the following: 1) Adversarial Testing: Creating robust testing frameworks to simulate challenging and adversarial user inputs, helping to identify vulnerabilities. 2) Error Handling: Improving the system's ability to recognize its limitations and gracefully handle ambiguous or incomplete queries by asking clarifying questions.3) Domain Generalization: Developing systems that can generalize effectively across different domains with minimal retraining or fine-tuning. 4) Noisy Input Handling: Enhancing the systems ability to process and respond to noisy or colloquial language, improving usability for diverse user demographics.

5. Efficiency

As dialogue systems become more complex, ensuring their efficiency in terms of computational resources, memory usage, and response time is crucial. Efficient systems can lower operational costs and provide faster responses, enhancing the user experience. Future research could include: 1) Model Optimization: Designing lightweight architectures and leveraging techniques like pruning, quantization, and distillation to reduce computational demands. 2) Energy Efficiency: Implementing green AI practices to minimize the energy consumption of training and deploying large-scale models. 3) On-Device Processing: Developing models that can operate efficiently on edge devices, reducing dependency on cloud-based infrastructure. 4) Few-Shot and Continual Learning: Exploring methods to reduce the need for large-scale retraining, enabling systems to adapt quickly to new tasks or domains with minimal data.

Chapter 9

Conclusion

In conclusion, this thesis has provided a comprehensive examination of various methodologies for integrating knowledge into task-oriented dialogue systems (TDS), highlighting both the opportunities and challenges associated with pipeline and end-to-end approaches. We have critically assessed the limitations of existing methods in integrating domain-specific knowledge, as well as common sense knowledge, into dialogue systems, a critical aspect for improving the systems ability to handle complex, real-world tasks. Through the exploration of schema-guided natural language understanding (NLU), relation-guided NLU, document-guided natural language generation (NLG), schema-guided user state modeling (USM), and large language model (LLM)-empowered end-to-end methods, we have demonstrated the potential of leveraging diverse forms of knowledge to substantially enhance system performance across various tasks.

Each of the methodologies discussed has shown distinct advantages in different scenarios, with schema-guided approaches providing structured and efficient ways to incorporate domain-specific knowledge, and LLM-empowered end-to-end models enabling more flexible and adaptive responses based on a broader understanding of language and context. The experimental evaluations presented throughout this thesis have validated these approaches, confirming their effectiveness in improving system accuracy, coherence, and relevance when interacting with users. These methodologies contribute to the continued advancement of task-oriented dialogue systems, offering valuable insights into how knowledge integration can drive more robust, context-aware, and user-friendly systems.

Looking ahead, the future of task-oriented dialogue systems lies in the development of more intelligent, adaptive, and user-centric systems that can efficiently assist users in completing tasks across a wide variety of domains and contexts. The evolution of such systems will require advancements in their ability to handle ambiguity, context shifts, and multi-turn dialogues with greater flexibility and accuracy. Addressing the research challenges outlined in this thesissuch as improving knowledge integration, enhancing reasoning capabilities, and increasing system transparencywill be crucial to achieving these goals. Moreover, with growing attention on the ethical implications of AI, future systems must be designed to ensure fairness, privacy, and accountability, aligning with broader societal values.

By focusing on these research directions, we can lay the groundwork for the next generation of task-oriented dialogue systemssystems that are not only

more capable and versatile but also ethically responsible and better equipped to meet the diverse needs of users. These advancements hold the potential to revolutionize how users interact with AI, enabling more personalized, efficient, and seamless assistance across a wide range of applications, from customer support to healthcare and beyond.

Bibliography

[1] Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. “Challenges in building intelligent open-domain dialog systems”. In: The ACM Transactions on Information Systems (TOIS), 2020.

[2] Hongshen Chen et al. “A survey on dialogue systems: Recent advances and new frontiers”. In: vol. 19. 2. ACM New York, NY, USA, 2017, pp. 25–35.

[3] Nikola Mrki et al. “Neural Belief Tracker: Data-Driven Dialogue State Tracking”. In: Association for Computational Linguistics (ACL), 2017.

[4] Chien-Sheng Wu et al. “Transferable multi-domain state generator for task-oriented dialogue systems”. In: Association for Computational Linguistics (ACL), 2019.

[5] Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. “Rethinking Action Spaces for Reinforcement Learning in End-to-end Dialog Agents with Latent Variable Models”. In: Proceedings of NAACL-HLT, 2019, pp. 1208–1218.

[6] Wenhu Chen et al. “Semantically Conditioned Dialog Response Generation via Hierarchical Disentangled Self-Attention”. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 3696–3709.

[7] Weiwei Sun et al. “Simulating user satisfaction for the evaluation of task-oriented dialogue systems”. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 2499–2506.

[8] Rishabh Mehrotra et al. “Jointly leveraging intent and interaction signals to predict user satisfaction with slate recommendations”. In: The World Wide Web Conference, 2019, pp. 1256–1267.

[9] Jianfeng Gao, Michel Galley, and Lihong Li. “Neural approaches to conversational AI”. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 1371–1374.

[10] Tsung-Hsien Wen et al. “A Network-based End-to-End Trainable Task-oriented Dialogue System”. In: Conference of the European Chapter of the Association for Computational Linguistics, 2017.

[11] Yue Feng, Yang Wang, and Hang Li. “A Sequence-to-Sequence Approach to Dialogue State Tracking”. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2021, pp. 1714–1725.

[12] Yue Feng et al. “Dynamic Schema Graph Fusion Network for Multi-Domain Dialogue State Tracking”. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 115–126.

[13] Yue Feng et al. “Schema-Guided User Satisfaction Modeling for Task-Oriented Dialogues”. In: arXiv preprint arXiv:2305.16798, 2023.

[14] Yue Feng et al. “Towards Asking Clarification Questions for Information Seeking on Task-Oriented Dialogues”. In: preprint arXiv:2305.13690, 2023.

[15] Yue Feng et al. “A large language model enhanced conversational recommender system”. In: arXiv preprint arXiv:2308.06212, 2023.

[16] K Yao et al. “Recurrent neural networks for language understanding”. In: ISCA. INTERSPEECH, Lyon, France, 2013, pp. 1–5.

[17] Kaisheng Yao et al. “Spoken language understanding using long short-term memory neural networks”. In: 2014 IEEE Spoken Language Technology Workshop (SLT).

[18] Dilek Hakkani-Tür et al. “Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM”. In: ISCA, 2016.

[19] Daniel Guo et al. “Joint semantic utterance classification and slot filling with recursive neural networks”. In: 2014 IEEE Spoken Language Technology Workshop (SLT).

[20] Puyang Xu and Ruhi Sarikaya. “Convolutional neural network based triangular crf for joint intent detection and slot filling”. In: IEEE. 2013 ieee workshop on automatic speech recognition and understanding, 2013, pp. 78–83.

[21] Kaisheng Yao et al. “Recurrent conditional random field for language understanding”. In: IEEE. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014, pp. 4077–4081.

[22] Jacob Devlin, Ming-Wei Chang, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: Proceedings of NAACL-HLT, 2019, pp. 4171–4186.

[23] Qian Chen, Zhu Zhuo, and Wen Wang. “Bert for joint intent classification and slot filling”. In: arXiv preprint arXiv:1902.10909, 2019.

[24] Giuseppe Castellucci et al. “Multi-lingual intent detection and slot filling in a joint bert-based model”. In: arXiv preprint arXiv:1907.02884, 2019.

[25] Chih-Wen Goo et al. “Slot-gated modeling for joint slot filling and intent prediction”. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics, 2018, pp. 753–757.

[26] Bing Liu and Ian Lane. “Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling”. In: Interspeech 2016, 2016, pp. 685–689.

[27] Steve Young et al. “Pomdp-based statistical spoken dialog systems: A review”. In: vol. 101. 5. IEEE, 2013, pp. 1160–1179.

[28] Jason D Williams and Steve Young. “Scaling POMDPs for spoken dialog management”. In: vol. 15. 7. IEEE, 2007, pp. 2116–2129.

[29] Jost Schatzmann et al. “Agenda-based user simulation for bootstrapping a POMDP dialogue system”. In: Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers, 2007, pp. 149–152.

[30] Matthew Henderson, Blaise Thomson, and Steve Young. “Word-based dialog state tracking with recurrent neural networks”. In: Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL), 2014, pp. 292–299.

[31] Nikola Mrki et al. “Multi-domain Dialog State Tracking using Recurrent Neural Networks”. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), 2015, pp. 794–799.

[32] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. “Sumbt: Slot-utterance matching for universal and scalable belief tracking”. In: Association for Computational Linguistics (ACL), 2019.

[33] Shuyang Gao et al. “Dialog state tracking: A neural reading comprehension approach”. In: Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 2019.

[34] Julien Perez. Machine reading method for dialog state tracking. US Patent 10,540,967. Jan. 2020.

[35] Jianguo Zhang et al. “Find or Classify? Dual Strategy for Slot-Value Predictions on Multi-Domain Dialog State Tracking”. In: Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics, 2020, pp. 154–167.

[36] Liliang Ren, Jianmo Ni, and Julian McAuley. “Scalable and accurate dialogue state tracking via hierarchical sequence generation”. In: Empirical Methods in Natural Language Processing (EMNLP), 2019.

[37] Li Zhou and Kevin Small. “Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering”. In: Neural Information Processing Systems (NeurIPS), 2019.

[38] Victor Zhong, Caiming Xiong, and Richard Socher. “Global-locally self-attentive encoder for dialogue state tracking”. In: Association for Computational Linguistics (ACL), 2018.

[39] Osman Ramadan, Pawe Budzianowski, and Milica Gasic. “Large-Scale Multi-Domain Belief Tracking with Knowledge Sharing”. In: Association for Computational Linguistics (ACL), 2018.

[40] Jian-Guo Zhang et al. “Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking”. In: Special Interest Group on Computational Semantics (SIGSEEM), 2020.

[41] Jiaying Hu et al. “Sas: Dialogue state tracking via slot attention and slot information sharing”. In: Association for Computational Linguistics (ACL), 2020.

[42] Shuyang Gao et al. “From machine reading comprehension to dialogue state tracking: Bridging the gap”. In: Association for Computational Linguistics (ACL), 2020.

[43] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2019.

[44] Alec Radford et al. “Language models are unsupervised multitask learners”. In: In OpenAI blog, 2019.

[45] Yichi Zhang et al. “A Probabilistic End-To-End Task-Oriented Dialog Model with Latent Belief States towards Semi-Supervised Learning”. In: Empirical Methods in Natural Language Processing (EMNLP), 2020.

[46] Zhaojiang Lin et al. “MinTL: Minimalist Transfer Learning for Task-Oriented Dialogue Systems”. In: Empirical Methods in Natural Language Processing (EMNLP), 2020.

[47] Weizhe Lin, Bo-Hsian Tseng, and Bill Byrne. “Knowledge-Aware Graph-Enhanced GPT-2 for Dialogue State Tracking”. In: Empirical Methods in Natural Language Processing (EMNLP), 2021.

[48] Lu Chen et al. “Schema-guided multi-domain dialogue state tracking with graph attention neural networks”. In: Association for the Advancement of Artificial Intelligence (AAAI), 2020.

[49] Su Zhu et al. “Efficient Context and Schema Fusion Networks for Multi-Domain Dialogue State Tracking”. In: Empirical Methods in Natural Language Processing (EMNLP), 2020.

[50] Yan Zeng and Jian-Yun Nie. “Multi-domain dialogue state tracking based on state graph”. In: arXiv preprint arXiv:2010.11137, 2020.

[51] Yawen Ouyang et al. “Dialogue state tracking with explicit slot connection modeling”. In: Association for Computational Linguistics (ACL), 2020.

[52] Yue Feng, Yang Wang, and Hang Li. “A Sequence-to-Sequence Approach to Dialogue State Tracking”. In: Association for Computational Linguistics (ACL), 2021.

[53] Michael Heck et al. “TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking”. In: Association for Computational Linguistics (ACL), 2020.

[54] Fanghua Ye et al. “Slot Self-Attentive Dialogue State Tracking”. In: The ACM Web Conference (WWW), 2021.

[55] Abhinav Rastogi et al. “Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset”. In: Association for the Advancement of Artificial Intelligence (AAAI), 2020.

[56] Vahid Noroozi et al. “A Fast and Robust BERT-based Dialogue State Tracker for Schema-Guided Dialogue Dataset”. In: arXiv preprint arXiv:2008.12335, 2020.

[57] Tsung-Hsien Wen et al. “Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems”. In: Association for Computational Linguistics. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

[58] Baolin Peng et al. “Few-shot Natural Language Generation for Task-Oriented Dialog”. In: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 172–182.

[59] Marilyn A Walker, Owen C Rambow, and Monica Rogati. “Training a sentence planner for spoken dialogue using boosting”. In: vol. 16. 3-4. Elsevier, 2002, pp. 409–433.

[60] Amanda Stent, Rashmi Prasad, and Marilyn Walker. “Trainable sentence planning for complex information presentations in spoken dialog systems”. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04), 2004, pp. 79–86.

[61] Alice H Oh and Alexander I Rudnicky. “Stochastic natural language generation for spoken dialog systems”. In: vol. 16. 3-4. Elsevier, 2002, pp. 387–407.

[62] Gabor Angeli, Percy Liang, and Dan Klein. “A simple domain-independent probabilistic approach to generation”. In: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, 2010, pp. 502–512.

[63] François Mairesse and Steve Young. “Stochastic language generation in dialogue using factored language models”. In: vol. 40. 4. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA publishers-info , 2014, pp. 763–799.

[64] Ravikumar Kondadadi, Blake Howald, and Frank Schilder. “A statistical nlg framework for aggregated planning and realization”. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, pp. 1406–1415.

[65] Tsung-Hsien Wen et al. “Multi-domain neural network language generation for spoken dialogue systems”. In: arXiv preprint arXiv:1603.01232, 2016.

[66] Van-Khanh Tran and Le-Minh Nguyen. “Natural language generation for spoken dialogue system using rnn encoder-decoder networks”. In: arXiv preprint arXiv:1706.00139, 2017.

[67] Shang-Yu Su et al. “Natural language generation by hierarchical decoding with linguistic patterns”. In: arXiv preprint arXiv:1808.02747, 2018.

[68] Pawe Budzianowski et al. “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, 2018, pp. 5016–5026.

[69] Shikib Mehri, Tejas Srinivasan, and Maxine Eskenazi. “Structured fusion networks for dialog”. In: arXiv preprint arXiv:1907.10016, 2019.

[70] Marilyn A Walker et al. “Generation and evaluation of user tailored responses in multimodal dialogue”. In: vol. 28. 5. Wiley Online Library, 2004, pp. 811–840.

[71] Srinivasan Janarthanam and Oliver Lemon. “Learning to adapt to unknown users: referring expression generation in spoken dialogue systems”. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010, pp. 69–78.

[72] Ondej Duek and Filip Juríek. “A context-aware natural language generator for dialogue systems”. In: arXiv preprint arXiv:1608.07076, 2016.

[73] François Mairesse and Marilyn A Walker. “Towards personality-based user adaptation: psychologically informed stylistic language generation”. In: vol. 20. Springer, 2010, pp. 227–278.

[74] Jelena Luketina et al. “A survey of reinforcement learning informed by natural language”. In: arXiv preprint arXiv:1906.03926, 2019.

[75] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: arXiv preprint arXiv:1609.08144, 2016.

[76] Dzmitry Bahdanau et al. “An actor-critic algorithm for sequence prediction”. In: arXiv preprint arXiv:1607.07086, 2016.

[77] Marc’Aurelio Ranzato et al. “Sequence level training with recurrent neural networks”. In: arXiv preprint arXiv:1511.06732, 2015.

[78] Yue Dong et al. “Banditsum: Extractive summarization as a contextual bandit”. In: arXiv preprint arXiv:1809.09672, 2018.

[79] Pradyumna Tambwekar et al. “Controllable neural story plot generation via reward shaping”. In: arXiv preprint arXiv:1809.10736, 2018.

[80] Julia Kreutzer et al. “Can neural machine translation be improved with user feedback?” In: arXiv preprint arXiv:1804.05958, 2018.

[81] Daniel M Ziegler et al. “Fine-tuning language models from human preferences. arXiv 2019”. In: arXiv preprint arXiv:1909.08593, 1909.

[82] Nisan Stiennon et al. “Learning to summarize with human feedback”. In: vol. 33. Advances in Neural Information Processing Systems, 2020, pp. 3008–3021.

[83] Braden Hancock et al. “Learning from dialogue after deployment: Feed yourself, chatbot!” In: arXiv preprint arXiv:1901.05415, 2019.

[84] Natasha Jaques et al. “Way off-policy batch deep reinforcement learning of implicit human preferences in dialog”. In: arXiv preprint arXiv:1-907.00456, 2019.

[85] Yongfeng Zhang et al. “Towards conversational search and recommendation: System ask, user respond”. In: Proceedings of the 27th acm international conference on information and knowledge management, 2018, pp. 177–186.

[86] Mohammad Aliannejadi et al. “Asking clarifying questions in open-domain information-seeking conversations”. In: Proceedings of the 42nd international acm sigir conference on research and development in information retrieval, 2019, pp. 475–484.

[87] Jingjing Xu et al. “Asking clarification questions in knowledge-based question answering”. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 1618–1629.

[88] Yang Trista Cao, Sudha Rao, and Hal Daumé III. “Controlling the Specificity of Clarification Question Generation”. In: Proceedings of the 2019 Workshop on Widening NLP, 2019, pp. 53–56.

[89] Sudha Rao and Hal Daumé III. “Answer-based Adversarial Training for Generating Clarification Questions”. 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), 2019, pp. 143–155.

[90] Vaibhav Kumar and Alan W Black. “ClarQ: A large-scale and diverse dataset for Clarification Question Generation”. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7296–7301.

[91] Mihail Eric et al. “MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines”. In: LREC, 2020.

[92] Fanghua Ye, Yue Feng, and Emine Yilmaz. “ASSIST: Towards Label Noise-Robust Dialogue State Tracking”. In: Findings of the Association for Computational Linguistics: ACL 2022, 2022, pp. 2719–2731.

[93] Yixuan Su et al. “Multi-Task Pre-Training for Plug-and-Play Task-Oriented Dialogue System”. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 4661–4676.

[94] Wenqiang Lei et al. “Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures”. In: Association for Computational Linguistics (ACL), 2018.

[95] Yue Feng, Gerasimos Lampouras, and Ignacio Iacobacci. “Topic-Aware Response Generation in Task-Oriented Dialogue with Unstructured Knowledge Access”. In: EMNLP, 2022.

[96] Libo Qin et al. “Dynamic Fusion Network for Multi-Domain End-to-end Task-Oriented Dialog”. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 6344–6354.

[97] Shiquan Yang, Rui Zhang, and Sarah Erfani. “GraphDialog: Integrating Graph Knowledge into End-to-End Task-Oriented Dialogue Systems”. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 1878–1888.

[98] Andrea Madotto, Chien-Sheng Wu, and Pascale Fung. “Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems”. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 1468–1478.

[99] Abhinav Rastogi et al. “Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset”. In: vol. 34. 05. Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 8689–8696.

[100] Kishore Papineni et al. “Bleu: a method for automatic evaluation of machine translation”. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[101] Chia-Wei Liu et al. “How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation”. In: EMNLP, 2016.

[102] Jan Deriu et al. “Survey on evaluation methods for dialogue systems”. In: vol. 54. 1. Springer, 2021, pp. 755–810.

[103] Shuo Zhang and Krisztian Balog. “Evaluating conversational recommender systems via user simulation”. In: Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining, 2020, pp. 1512–1520.

[104] Wieland Eckert, Esther Levin, and Roberto Pieraccini. “User modeling for spoken dialogue system evaluation”. In: IEEE. 1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings, 1997, pp. 80–87.

[105] Steve Young and K Sheffler. “Probabilistic Simulation of Human-Machine Dialogues”. In: Proc. ICASSP, Istanbul, Turkey, 2000.

[106] Kallirroi Georgila, James Henderson, and Oliver Lemon. “Learning user simulations for information state update dialogue systems.” In: Interspeech, 2005, pp. 893–896.

[107] Heriberto Cuayáhuitl et al. “Human-computer dialogue simulation using hidden markov models”. In: IEEE. IEEE Workshop on Automatic Speech Recognition and Understanding, 2005., 2005, pp. 290–295.

[108] Layla El Asri, Jing He, and Kaheer Suleman. “A sequence-to-sequence model for user simulation in spoken dialogue systems”. In: arXiv preprint arXiv:1607.00070, 2016.

[109] Florian Kreyssig et al. “Neural user simulation for corpus-based policy optimisation for spoken dialogue systems”. In: arXiv preprint arXiv:1805.06966, 2018.

[110] Weiwei Sun et al. “Metaphorical user simulators for evaluating task-oriented dialogue systems”. In: vol. 42. 1. ACM New York, NY, 2023, pp. 1–29.

[111] To Eun Kim and Aldo Lipani. “A Multi-Task Based Neural Model to Simulate Users in Goal-Oriented Dialogue Systems”. In: SIGIR. 2022.

[112] Fanghua Ye, Zhiyuan Hu, and Emine Yilmaz. “Modeling user satisfaction dynamics in dialogue via hawkes process”. In: arXiv preprint arXiv:2305.12594, 2023.

[113] Zhiyuan Hu et al. “Unlocking the potential of user feedback: Leveraging large language model as user simulators to enhance dialogue system”. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 3953–3957.

[114] Yang Deng et al. “User Satisfaction Estimation with Sequential Dialogue Act Modeling in Goal-oriented Conversational Systems”. In: Proceedings of the ACM Web Conference 2022, 2022, pp. 2998–3008.

[115] Liyi Guo et al. “A Deep Prediction Network for Understanding Advertiser Intent and Satisfaction”. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 2501–2508.

[116] Zhijing Wu et al. “The influence of image search intents on user behavior and satisfaction”. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 645–653.

[117] Ning Su et al. “User intent, behaviour, and perceived satisfaction in product search”. In: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 547–555.

[118] Rishabh Mehrotra et al. “User interaction sequences for search satisfaction prediction”. In: Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval, 2017, pp. 165–174.

[119] Seyyed Hadi Hashemi et al. “Measuring user satisfaction on smart speaker intelligent assistants using intent sensitive query embeddings”. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 1183–1192.

[120] Kaisong Song et al. “Using Customer Service Dialogues for Satisfaction Analysis with Context-Assisted Multiple Instance Learning”. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2019, pp. 198–207.

[121] Klaus-Peter Engelbrecht et al. “Modeling user satisfaction with hidden Markov models”. In: Proceedings of the SIGDIAL 2009 Conference, 2009, pp. 170–177.

[122] Praveen Kumar Bodigutla et al. “Joint Turn and Dialogue level User Satisfaction Estimation on Multi-Domain Conversations”. In: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 3897–3909.

[123] Zhaohao Zeng et al. “Overview of the ntcir-15 dialogue evaluation (dialeval-1) task”. In: 2020.

[124] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: vol. 521. 7553. Nature Publishing Group UK London, 2015, pp. 436–444.

[125] Antoine Bordes, Y-Lan Boureau, and Jason Weston. “Learning End-to-End Goal-Oriented Dialog”. In: International Conference on Learning Representations, 2016.

[126] Mihail Eric et al. “Key-Value Retrieval Networks for Task-Oriented Dialogue”. In: Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, 2017, pp. 37–49.

[127] Bhuwan Dhingra et al. “Towards End-to-End Reinforcement Learning of Dialogue Agents for Information Access”. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 484–495.

[128] Xiujun Li et al. “End-to-End Task-Completion Neural Dialogue Systems”. In: Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2017, pp. 733–743.

[129] Bing Liu and Ian Lane. “Iterative policy learning in end-to-end trainable task-oriented neural dialog models”. In: IEEE. 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2017, pp. 482–489.

[130] Tiancheng Zhao and Maxine Eskenazi. “Towards End-to-End Learning for Dialog State Tracking and Management using Deep Reinforcement Learning”. In: Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2016, pp. 1–10.

[131] Jason D Williams, Kavosh Asadi Atui, and Geoffrey Zweig. “Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning”. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 665–677.

[132] Chenzhan Shang et al. “Multi-grained Hypergraph Interest Modeling for Conversational Recommendation”. In: arXiv preprint arXiv:2305.04798, 2023.

[133] Kun Zhou et al. “Leveraging historical interaction data for improving conversational recommender system”. In: Proceedings of the 29th ACM international conference on information & knowledge management, 2020, pp. 2349–2352.

[134] Wenqiang Lei et al. “Estimation-action-reflection: Towards deep interaction between conversational and recommender systems”. In: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 304–312.

[135] Xuhui Ren et al. “Learning to ask appropriate questions in conversational recommendation”. In: Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval, 2021, pp. 808–817.

[136] Kerui Xu et al. “Adapting user preference to online feedback in multi-round conversational recommendation”. In: Proceedings of the 14th ACM international conference on web search and data mining, 2021, pp. 364–372.

[137] Raymond Li et al. “Towards deep conversational recommendations”. In: vol. 31. Advances in neural information processing systems, 2018.

[138] Jinfeng Zhou et al. “CRFR: Improving conversational recommender systems via flexible fragments reasoning on knowledge graphs”. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 4324–4334.

[139] Qibin Chen et al. “Towards Knowledge-Based Recommender Dialog System”. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019, pp. 1803–1813.

[140] Kun Zhou et al. “Towards Topic-Guided Conversational Recommender System”. In: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 4128–4139.

[141] Kun Zhou et al. “Improving conversational recommender systems via knowledge graph based semantic fusion”. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining, 2020, pp. 1006–1014.

[142] Quan Tu et al. “Conversational recommendation via hierarchical information modeling”. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 2201–2205.

[143] Yuanhang Zhou et al. “C^2-CRS: Coarse-to-Fine Contrastive Learning for Conversational Recommender System”. In: Proceedings of WSDM, 2022, pp. 1488–1496.

[144] Likang Wu et al. “A Survey on Large Language Models for Recommendation”. In: arXiv preprint arXiv:2305.19860, 2023.

[145] Peng Liu, Lemei Zhang, and Jon Atle Gulla. “Pre-train, prompt and recommendation: A comprehensive survey of language modelling paradigm adaptations in recommender systems”. In: arXiv preprint arXiv:2302.03735, 2023.

[146] Xiaolei Wang et al. “Rethinking the Evaluation for Conversational Recommendation in the Era of Large Language Models”. In: arXiv preprint arXiv:2305.13112, 2023.

[147] Kimiya Keyvan and Jimmy Xiangji Huang. “How to approach ambiguous queries in conversational search: A survey of techniques, approaches, tools, and challenges”. In: 55.6 (2022), pp. 1–40.

[148] Charles LA Clarke, Nick Craswell, and Ian Soboroff. “Overview of the TREC 2009 Web Track.” In: vol. 9. Trec, 2009, pp. 20–29.

[149] David Schlangen. “Causes and strategies for requesting clarification in dialogue”. In: Proceedings of the 5th SIGdial Workshop on Discourse and Dialogue at HLT-NAACL 2004, 2004, pp. 136–143.

[150] Corbin Rosset et al. “Leading conversational search by suggesting useful questions”. In: Proceedings of the web conference 2020, 2020, pp. 1160–1170.

[151] Alessandro Sordoni et al. “A hierarchical recurrent encoder-decoder for generative context-aware query suggestion”. In: proceedings of the 24th ACM international on conference on information and knowledge management, 2015, pp. 553–562.

[152] Liu Yang et al. “Neural matching models for question retrieval and next question prediction in conversation”. In: (2017).

[153] Keping Bi, Qingyao Ai, and W Bruce Croft. “Asking clarifying questions based on negative feedback in conversational search”. In: (2021), pp. 157–166.

[154] Sudha Rao and Hal Daumé III. “Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information”. In: (2018).

[155] Hamed Zamani et al. “Generating clarifying questions for information retrieval”. In: Proceedings of the web conference 2020. 2020, pp. 418–428.

[156] Filip Radlinski and Nick Craswell. “A theoretical framework for conversational search”. In: Proceedings of the 2017 conference on conference human information interaction and retrieval, 2017, pp. 117–126.

[157] Alexandra Vtyurina et al. “Exploring conversational search with humans, assistants, and wizards”. In: Proceedings of the 2017 chi conference extended abstracts on human factors in computing systems, 2017, pp. 2187–2193.

[158] Mohammad Aliannejadi et al. “ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ)”. In: (2020).

[159] Kaustubh D Dhole. “Resolving intent ambiguities by retrieving discriminative clarifying questions”. In: (2020).

[160] Jian Wang and Wenjie Li. “Template-guided clarifying question generation for web search clarification”. In: Proceedings of the 30th ACM international conference on information & knowledge management, 2021, pp. 3468–3472.

[161] Ivan Sekuli, Mohammad Aliannejadi, and Fabio Crestani. “Towards facet-driven generation of clarifying questions for conversational search”. In: Proceedings of the 2021 ACM SIGIR international conference on theory of information retrieval, 2021, pp. 167–175.

[162] Ziliang Zhao et al. “Generating clarifying questions with web search results”. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 234–244.

[163] Nikola Mrki et al. “Neural belief tracker: Data-driven dialogue state tracking”. In: In ACL, 2017.

[164] Xiaoxue Zang et al. “MultiWOZ 2.2: A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines”. In: Association for Computational Linguistics (ACL), 2020.

[165] Zhi Chen et al. “CREDIT: Coarse-to-Fine Sequence Generation for Dialogue State Tracking”. In: arXiv preprint arXiv:2009.10435, 2020.

[166] Ehsan Hosseini-Asl et al. “A simple language model for task-oriented dialogue”. In: arXiv preprint arXiv:2005.00796, 2020.

[167] Giovanni Paolini et al. “Structured Prediction as Translation between Augmented Natural Languages”. In: In ICLR, 2021.

[168] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. “Pointer networks”. In: In NIPS, 2015.

[169] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: In ICLR, 2015.

[170] Tsung-Hsien Wen et al. “A network-based end-to-end trainable task-oriented dialogue system”. In: In EACL, 2017.

[171] Matthew Henderson, Blaise Thomson, and Jason D Williams. “The second dialog state tracking challenge”. In: In SIGDIAL, 2014.

[172] Pararth Shah et al. “Building a conversational agent overnight with dialogue self-play”. In: arXiv preprint arXiv:1801.04871, 2018.

[173] Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. “What is left to be understood in ATIS?” In: In SLT, 2010.

[174] Alice Coucke et al. “Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces”. In: In CoRR, 2018.

[175] Guan-Lin Chao and Ian Lane. “Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer”. In: In INTERSPEECH, 2019.

[176] Liliang Ren et al. “Towards universal dialogue state tracking”. In: Empirical Methods in Natural Language Processing (EMNLP), 2018.

[177] Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. “Multi-task Learning for Joint Language Understanding and Dialogue State Tracking”. In: Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL), 2018.

[178] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: The International Conference on Learning Representations (ICLR), 2014.

[179] Zheng Zhang et al. “Recent advances and challenges in task-oriented dialog systems”. In: In Science China Technological Sciences, 2020.

[180] Bernardino Romera-Paredes and Philip HS Torr. “An embarrassingly simple approach to zero-shot learning”. In: International Conference on Machine Learning (ICML), 2015.

[181] Petar Velikovi et al. “Graph attention networks”. In: The International Conference on Learning Representations (ICLR), 2018.

[182] Ashish Vaswani et al. “Attention is All you Need”. In: Neural Information Processing Systems (NeurIPS), 2017.

[183] Xiao Wang et al. “Heterogeneous graph attention network”. In: The ACM Web Conference (WWW), 2019.

[184] Mihail Eric et al. “Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines”. In: The International Conference on Language Resources and Evaluation (LREC), 2020.

[185] Sungdong Kim et al. “Efficient Dialogue State Tracking by Selectively Overwriting Memory”. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020.

[186] Yixuan Su et al. “Multi-Task Pre-Training for Plug-and-Play Task-Oriented Dialogue System”. In: arXiv preprint arXiv:2109.14739, 2021.

[187] Wanwei He et al. “Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection”. In: vol. 36. 10. Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 10749–10757.

[188] Andrea Madotto et al. “Learning Knowledge Bases with Parameters for Task-Oriented Dialogue Systems”. In: Findings of the Association for Computational Linguistics: EMNLP 2020, 2020, pp. 2372–2394.

[189] Mohammad Aliannejadi et al. “Building and Evaluating Open-Domain Dialogue Corpora with Clarifying Questions”. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 4473–4484.

[190] Song Feng et al. “doc2dial: A Goal-Oriented Document-Grounded Dialogue Dataset”. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 8118–8128.

[191] Rishabh Mehrotra and Emine Yilmaz. “Terms, topics & tasks: Enhanced user modelling for better personalization”. In: Proceedings of the 2015 international conference on the theory of information retrieval, 2015, pp. 131–140.

[192] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7871–7880.

[193] Marzieh Saeidi et al. “Interpretation of Natural Language Rules in Conversational Machine Reading”. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2018, pp. 2087–2097.

[194] Hanxun Zhong et al. “Less is More: Learning to Refine Dialogue History for Personalized Dialogue Generation”. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2022, pp. 5808–5820.

[195] Chien-Sheng Wu et al. “Getting To Know You: User Attribute Extraction from Dialogues”. In: Proceedings of the 12th Language Resources and Evaluation Conference, 2020, pp. 581–589.

[196] Jiahuan Pei, Pengjie Ren, and Maarten de Rijke. “A cooperative memory network for personalized task-oriented dialogue systems with incomplete user profiles”. In: Proceedings of the Web Conference 2021, 2021, pp. 1552–1561.

[197] Hongjin Qian et al. “Learning implicit user profile for personalized retrieval-based chatbot”. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 1467–1477.

[198] Baolin Peng et al. “SOLOIST: Building Task Bots at Scale with Transfer Learning and Machine Teaching”. In: vol. 9. Transactions of the Association for Computational Linguistics, 2021, pp. 907–824.

[199] Yunyi Yang, Yunhao Li, and Xiaojun Quan. “UBAR: Towards Fully End-to-End Task-Oriented Dialog System with GPT-2”. In: vol. 35. 16. Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 14230–14238.

[200] Bodhisattwa Prasad Majumder et al. “Ask whats missing and whats useful: Improving Clarification Question Generation using Global Knowledge”. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2021, pp. 4300–4312.

[201] Kishore Papineni et al. “Bleu: a method for automatic evaluation of machine translation”. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[202] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: Text summarization branches out, 2004, pp. 74–81.

[203] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: Text summarization branches out, 2004, pp. 74–81.

[204] Mohammad Kachuee et al. “Self-Supervised Contrastive Learning for Efficient User Satisfaction Prediction in Conversational Agents”. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics, 2021, pp. 4053–4064.

[205] Mika Rebensburg, Stefan Hillmann, and Nils Feldhus. “AUTOMATIC USER EXPERIENCE EVALUATION OF GOAL-ORIENTED DIALOGS USING PRE-TRAINED LANGUAGE MODELS”. In: In Proc. ESSV 2023 (March 13, Munich), TUDpress., 2023.

[206] Jaime Carbonell and Jade Goldstein. “The use of MMR, diversity-based reranking for reordering documents and producing summaries”. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, 1998, pp. 335–336.

[207] Trevor Hastie et al. “The elements of statistical learning: data mining, inference, and prediction”. In: vol. 2. Springer, 2009.

[208] Meng Chen et al. “The JDDC Corpus: A Large-Scale Multi-Turn Chinese Dialogue Dataset for E-commerce Customer Service”. In: LREC, 2020.

[209] Wenxiang Jiao et al. “HiGRU: Hierarchical Gated Recurrent Units for Utterance-Level Emotion Recognition”. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, 2019, pp. 397–406.

[210] Rahul Dey and Fathi M Salem. “Gate-variants of gated recurrent unit (GRU) neural networks”. In: IEEE. 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), 2017, pp. 1597–1600.

[211] Zichao Yang et al. “Hierarchical attention networks for document classification”. In: Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics, 2016, pp. 1480–1489.

[212] Wanling Cai and Li Chen. “Predicting user intents and satisfaction with dialogue-based conversational recommendations”. In: Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization, 2020, pp. 33–42.

[213] Jason Ingyu Choi, Ali Ahmadvand, and Eugene Agichtein. “Offline and online satisfaction prediction in open-domain conversational systems”. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1281–1290.

[214] Chongming Gao et al. “Advances and challenges in conversational recommender systems: A survey”. In: vol. 2. Elsevier, 2021, pp. 100–126.

[215] Tom Brown et al. “Language models are few-shot learners”. In: vol. 33. Advances in neural information processing systems, 2020, pp. 1877–1901.

[216] Long Ouyang et al. “Training language models to follow instructions with human feedback”. In: vol. 35. Advances in Neural Information Processing Systems, 2022, pp. 27730–27744.

[217] Aakanksha Chowdhery et al. “Palm: Scaling language modeling with pathways”. In: arXiv preprint arXiv:2204.02311, 2022.

[218] Aohan Zeng et al. “Glm-130b: An open bilingual pre-trained model”. In: ICLR, 2022.

[219] Susan Zhang et al. “Opt: Open pre-trained transformer language models”. In: arXiv preprint arXiv:2205.01068, 2022.

[220] Hugo Touvron et al. “Llama: Open and efficient foundation language models”. In: arXiv preprint arXiv:2302.13971, 2023.

[221] Yunfan Gao et al. “Chat-rec: Towards interactive and explainable llms-augmented recommender system”. In: arXiv preprint arXiv:2303.14524, 2023.

[222] Timo Schick et al. “Toolformer: Language models can teach themselves to use tools”. In: arXiv preprint arXiv:2302.04761, 2023.

[223] Yongliang Shen et al. “Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface”. In: arXiv preprint arXiv:2303.17580, 2023.

[224] Yuxi Li. “Deep reinforcement learning: An overview”. In: arXiv preprint arXiv:1701.07274, 2017.

[225] Shayne Longpre et al. “The flan collection: Designing data and methods for effective instruction tuning”. In: arXiv preprint arXiv:2301.13688, 2023.

[226] Shuzi Niu et al. “Top-k learning to rank: labeling, ranking and evaluation”. In: Proceedings of SIGIR, 2012, pp. 751–760.

[227] Richard S Sutton and Andrew G Barto. “Reinforcement learning: An introduction”. In: MIT press, 2018.

[228] Dongyeop Kang et al. “Recommendation as a Communication Game: Self-Supervised Bot-Play for Goal-oriented Dialogue”. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, 2019, pp. 1951–1961.

[229] Alexander Miller et al. “ParlAI: A Dialog Research Software Platform”. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2017, pp. 79–84.

[230] Alexander Miller et al. “Key-Value Memory Networks for Directly Reading Documents”. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016, pp. 1400–1409.

[231] Yu Wu et al. “Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-Based Chatbots”. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 496–505.

[232] Ellen M Voorhees and L Buckland. “Overview of the TREC 2003 Question Answering Track.” In: vol. 2003. TREC, 2003, pp. 54–68.

[233] Kalervo Järvelin and Jaana Kekäläinen. “IR evaluation methods for retrieving highly relevant documents”. In: vol. 51. 2. ACM SIGIR Forum, 2017, pp. 243–250.

[234] Jiwei Li et al. “A Diversity-Promoting Objective Function for Neural Conversation Models”. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 110–119.

[235] Kun Zhou et al. “CRSLab: An Open-Source Toolkit for Building Conversational Recommender System”. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 2021, pp. 185–193.

bibliography