Doctoral Dissertation

Zero-shot slot filling based on intrinsic representations from multiple aspects

Li Sixia

Supervisor Okada Shogo

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
Information Science

March, 2023

Abstract

Building dialogue systems that can smoothly communicate with humans is an enduring topic in artificial intelligence. Nowadays, the task-oriented dialogue system is widely used in real-world business to help users complete specific tasks. In the task-oriented dialogue system, the pipeline system is a popular solution so far. Although the end-to-end system was shown as good performance as the pipeline system recently, the pipeline system is advantageable on that each module of the pipeline system can be separately analyzed and studied. This makes the pipeline system more suitable on researching specific problems. Considering this advantage for research, our research focuses on the pipeline system.

With the developments of services and applications, an existing system is required to be extended to new domains that describe new tasks and topics. In adapting to new domains, conventional systems often needed to retrain the model to handle new task-related classes in new domains, which was inefficient in practice. For this problem, one of the ultimate goals of the task-oriented dialogue system was proposed, that is to build a domain-adaptable system that could be adapted to any given domain without training instances, so-called zero-shot domain adaptation.

In the pipeline system, the slot filling module is the immediate module connecting the user and the system. Slot filling aims to extract the user's intended components by predicting slot entities and slot types. The slot entity indicates the tokens that belong to a slot. The slot type indicates the specific slot that the slot entity is belonging to. The intended components carry the information about completing the task. These components are the basis for the subsequential modules' processes. Therefore, the zero-shot capacity of the entire dialogue system relies on the zero-shot capacity of the slot filling module a lot.

Towards the ultimate goal, conventional slot filling methods were inefficient since they needed to retrain to handle unseen slots in new domains. Zero-shot slot filling was proposed to deal with this problem. Zero-shot slot filling aims to train a model on source domains and adapt the model on target domains directly. Previous zero-shot slot filling methods handled target domains mainly relying the domain similarities based on explicit information. However, the improvements in zero-shot slot filling were limited. The main reason is because the explicit information is sensitive to domain shift problems. Specifically, domain shift problems influence the model

performance from three aspects. The first one is the unseen slots that were not appeared in training domains can be encountered in new domains. The second one is seen slots can be differently explained due to the topic changes in new domains. The third one is the context distributions is generally different in new domains. Due to these domain shift problems, it is hard to treat new domains based on the knowledge learned in training.

The overall objective of our research is to mine intrinsic representations that describe intrinsic characteristics of slots and values to alleviate domain shift problems in zero-shot slot filling towards the ultimate goal. Generally, as the intrinsic characteristics of an object could be stable whatever its specific appearance is, the intrinsic characteristics of slots and values could be expected to be more common across domains, thus providing effective transferable information. Specifically, we separately mined intrinsic representations from three aspects to alleviate domain shift problems in zero-shot slot filling. These representations are the inference relation path (IRP), the multi-relation-based representation, and the ontology-based representation.

We proposed the IRP from the knowledge graph to deal with the domain shift problem of unseen slots. We conducted a statistical analysis and showed that IRPs implicitly carry the relationships between slots and the values of specific meanings. Such relationships were not domain dependent. Thus IRPs could be expected to provide transferable information across domains to alleviate domain shift problems. Experimental results of utilizing IRPs in zero-shot slot filling demonstrated that using IRPs improved zero-shot slot filling by alleviating domain shift problems, especially on the unseen slots. However, IRPs were not flexible to be used since the extraction of IRPs needed the slots and values to be identified as entities, while many slots and values could not be identified as entities in practice. Moreover, the absolute improvement by using IRPs on zero-shot slot filling was not high.

To overcome the limitations of IRPs to alleviate domain shift problems, we proposed the multi-relation-based representation to deal with the domain shift problem of different context distributions. The multi-relation-based representation captures general meanwhile specific characteristic of slot entities among a variety of context environments and slots. Thus it could be expected to provide effective transferable information across domains. Experimental results and analysis demonstrated that the proposed representation alleviated the domain shift problem on the slot entity prediction, thus improving zero-shot slot filling. However, improving slot entity predictions alone could not improve zero-shot slot filling for unseen slots much due to the limitations in the slot type prediction.

To fill the gap of lacking knowledge in slot type predictions to handle domain shift problems of unseen slots and differently explained seen slots,

we proposed the ontology-based representation for the slot type prediction. The ontology is a pre-defined knowledge base that describes the intrinsic relationships between slots and their values. In our research, we assume the ontology for each new domain is fully defined, which contains all slots and possible values. When a domain changes from one to another, the definitions in the ontology will not change. Thus the ontology could establish the relationships between slots and values across domains to handle the slots in new domains. Experimental results and analysis demonstrated that using the ontology-based representation significantly improved zero-shot slot filling. We combined the use of the multi-relation-based representation with the ontology-based representation and showed further alleviation of domain shift problems in zero-shot slot filling.

Finally, we investigated whether the improvements of zero-shot slot filling contribute to the performance of the task-oriented dialogue system, which was unclear in previous studies. We compared dialogue systems using different slot filling modules, including the modules based on conventional methods and the modules based on zero-shot methods. Experimental results demonstrated that zero-shot methods generally contribute to improve the zero-shot capacity of the task-oriented dialogue system from different aspects when encountering unseen domains.

In conclusion, by alleviating the domain shift problems in zero-shot slot filling, the proposed intrinsic representations were effective towards the ultimate goal of the task-oriented dialogue system.

Keywords: Task-oriented dialogue system, zero-shot slot filling, neural network, knowledge graph, ontology, machine learning

Acknowledgment

First and foremost, I would like to express my sincere gratitude to my research supervisors, Prof. Jianwu Dang and Prof. Shogo Okada. Thanks to Prof. Jianwu Dang for his patient guidance and strict teaching. Thanks to Prof. Shogo Okada for his patient correction and careful guidance. I was able to learn how to carry out research and how to make good research. This dissertation would never be completed without my research supervisors' supports.

Secondly, I would like to express my sincere gratitude to my minor research supervisors, Prof. Masato Akagi and Prof. Ryuhei Uehara. Thanks for their helps to make my research process smoother.

And I would like to appreciate to my lab mates and friends. In those years in JAIST, many people helped me a lot on research and life. Thanks to them for making my research life more interesting.

Last but not least, I appreciate the research facilities and financial support for my research. I would like to express my gratitude for the Doctoral Research Fellow (DRF) from JAIST, and the grants that supported my research.

Acronym and Abbreviations

BiLSTM	Bi-directional long-short term memory network
Coach	Coarse-to-fine approache for zero-shot slot filling
CRF	Conditional random field
CT	Concept tagger network
CZSL-adv	Contrastive Zero-Shot Learning with adversarial attack
DP	Dialogue policy management
DST	Dialogue state tracking
IRP	Inference relation path
LSTM	Long-short term memory network
MD slots	Multi-domain slots
NLG	Natural language generation
NLU	Natural language understanding
PCLC	Prototypical contrastive learning and label confusion network
RNN	Recurrent neural network
RZT	Robust zero-shot tagger with examples
SD slots	Single-domain slots
ZAT	Zero-shot domain adaptive transfer network

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Research Background

Dialogue is a fundamental communication approach for human beings. We exchange information, emotions, and opinions through dialogues. In the research area of computer science, developing an automatic dialogue system that can make dialogues like humans is an enduring topic. Since the ELIZA [1] system in the 1960s, various dialogue systems have been developed. From rule-based systems [1, 2] to frame-based systems [3, 4], from statistical method-based systems [5] to deep neural network-based and pre-trained-based systems [6, 7, 8, 9].

Dialogue systems can be widely seen in daily life. Based on the primary function, dialogue systems can be divided into two categories: the task-oriented dialogue system and the non-task-oriented system, which is also called the chatbot system. The task-oriented dialogue system focuses on satisfying users' requests to complete specific tasks, such as booking hotels and trains, searching for requested music and movies, and showing weathers. The chatbot system focuses on making smooth and natural dialogues to engage users. Nowadays, task-oriented dialogue systems are widely used in the real world, such as customer services and smart applications.

Due to the requirements of task-oriented dialogue systems in real-world business, building task-oriented dialogue systems has become a hot topic in industry and research areas [10]. In the task-oriented dialogue system, the pipeline structure is a popular approach. A pipeline system generally consists of multiple modules, each module has its own function on handling dialogues. Besides pipeline systems, the end-to-end system [13, 69] that aims to generate responses from the input utterances directly has also attracted much attention to streamline the system process. So far, the pipeline system was considered better performance than end-to-end systems [14]. Recently, a fully end-to-end system was shown as high performance as the pipeline system [13], although it was in a different evaluation setting to [14]. Accordingly, pipeline systems and end-to-end systems have their advantages on handling

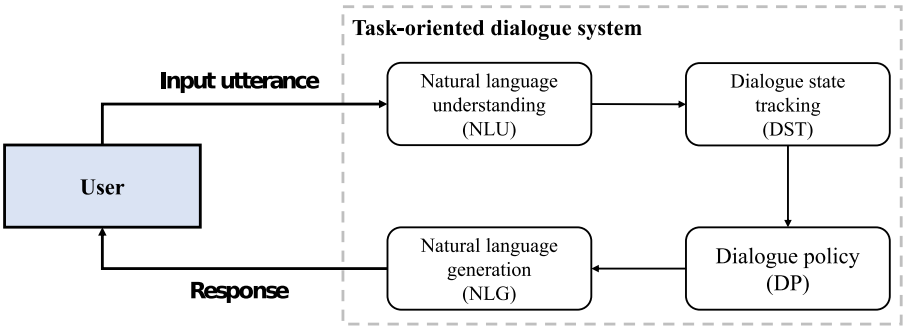

Figure 1.1: The structure of the task-oriented dialogue system

different situations. Nonetheless, the pipeline system has an advantage that the end-to-end system has not, that is each module of the pipeline system can be separately analyzed and studied. This makes the pipeline system more suitable on researching specific problems. The advantage of the pipeline system also makes the system more reliable in the industry area. For analyzing and solving problems that will be decribed later, our research focuses on the pipeline system.

Generally, the pipeline task-oriented dialogue system consisted of four modules. Figure 1.1 shows the modules and the system's interaction process with users. When the user gives an utterance to the system, the natural language understanding (NLU) extracts the intended components by the slot filling module and recognizes the user's intention by the intention detection module. The intended components are the basis for subsequential modules since those components carry the contents and requests of the user's utterance. After the NLU, the dialogue state tracking (DST) module receives the semantic components and updates the current dialogue states. Those dialogue states record the information about the task from the beginning of the dialogue. Next, the dialogue policy (DP) module select response actions based on the current dialogue states. Finally, the natural language generation (NLG) module generates the response to the user.

With the extension and development of applications, a task-oriented dialogue system is required to handle increasing scenes and to be generalizable to deal with diverse domains. The domain here indicates a specific topic or task, such as booking hotels, asking weather, and setting up services. According to these requirements, researchers pointed out one of the ultimate goals of the task-oriented dialogue system: that is to build a domain-adaptable system that can be adapted to any given domain without training instances [10], so-called zero-shot domain adaptation. Towards this ultimate goal, the task-oriented dialogue system is required to be zero-shot adaptable to new

domains, which means a system must handle unseen classes, such as unseen slots in new domains.

As described above, in the task-oriented dialogue system, slot filling extracts the intended components from given utterances. The correctness of those components influences the accuracy of the subsequential processes. Therefore, whether the slot filling module could deal well with the slots in new domains determines the system's performance when extending to new domains. Accordingly, the zero-shot capacity of slot filling influences the zero-shot capacity of the task-oriented dialogue system a lot. Improving the zero-shot domain adaptation capacity for slot filling is an important task towards the ultimate goal of the task-oriented dialogue system.

In slot filling, a slot is usually a predefined unit for describing a specific semantic meaning, topic, or category of an intended component in a specific domain. The intended component that belongs to a specific slot is generally called the value of the slot. For example, in the domain of booking hotels, the slot 'stay people' can be predefined for describing the number of people that want to stay. The values of 'stay people' could include '1'and '2.' The slots with the same name can express different meanings in different domains. For example, the slot 'object name' describes book names in the domain of rating books, while this slot describes movie names in the domain of searching creative works.

As the mission of slot filling is to extract the intended components from the user's utterances and fill the components into specific slots, slot filling is usually realized as a sequence tagging task. Specifically, slot filling aims to tag each token in the given utterance to indicate two components: the slot entity and the slot type. The slot entity is an entity that belongs to a slot, which is usually consisted of several tokens. The slot entity is equivalent to the slot value in describing a specific entity that belonging to a slot. But the slot entity is usually used in the context that identifying the entity, the slot value is usually used in the context of describing the relationship between the slot and the value. The slot type is the specific type that the slot entity is belonging to, which is usually described by the slot name. The slot entity is usually indicated by the inner-outer-beginning (IOB) tags. The I and B tags indicate that the corresponding token is the inner or the beginning of a slot; the O tag means that the corresponding token is out of any slot. The slot type is usually indicated by the slot type (slot name) tag. In conventional slot filling approaches, the IOB tag and the slot type tag was usually used in combination of IOB-slot type tags to indicate the slot entity and the slot type for each token. Figure 1.2 shows an example of slot filling using the IOB-slot tagging system. As seen in the figure, for the given utterance "Will it be windy at 4 pm in NY?" the slot filling module tags windy, 4 pm,

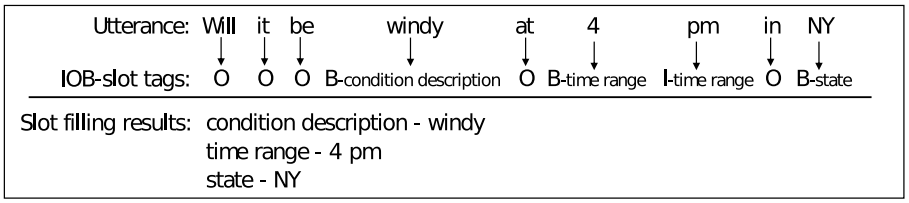

Figure 1.2: Example of IOB-slot type tagging results by slot filling

and NY as B-condition description, B-time range, I-time range, and B-state, respectively. These tags indicate that the windy, 4 pm, and NY are slot entities and belong to the slot types of 'condition description', 'time range,' and 'state,' respectively. The slot filling module tags other tokens as O, which means other tokens are out of any slot. As a result, the slot filling module extract 'condition description: windy,' 'time range: 4 pm', and 'state: NY' as the intended components and convey those components to subsequential modules.

Towards the ultimate goal of the task-oriented dialogue system, the slot filling module will encounter unseen slots that were not appeared in training and the seen slots with different explanations when the domain shifts from one to another. Moreover, the context environments in new domains may change much than in training domains. However, conventional slot filling methods are unsuitable for those problems. The main limitation is that conventional methods cannot handle unseen slot classes without modifying the parameters of the classification layer, so they fail to deal with unseen domains.

Zero-shot slot filling was proposed to deal with unseen slots in new domains. Zero-shot slot filling aims to train a model on source domains and directly adapt the model on target domains without training instances of target domains. The domain adaptation relies on the implicit similarities between domains. These similarities can be reflected on many aspects, such as slots and contexts. To handle any given domain containing different slots, zero-shot slot filling methods generally predict slot entities and slot types separately to avoid retraining the model for unseen slots. Specifically, two approaches were used to realize zero-shot slot filling, the one-stage approaches and the two-stage pipeline approach. The one-stage approach predicted slot entities for each given slot type directly. The two-stage approach generally used two stages by predicting slot entities in stage one and predict slot types for each slot entity in stage two.

1.2 Research Motivation

Zero-shot slot filling is a challenging task so far since no training instances of new domains are given. No training instances leads to the situation that new domains are difficult to be handled by the knowledge learned in training. To date, previous studies on zero-shot slot filling mainly focused on developing new models or algorithms on utilizing the similarities between domains [19, 20, 21] and improving the predictions of slot entity [22] and unseen slots [23]. However, the improvements so far were limited. The main reason was that previous studies mainly relied on the domain similarities based on explicit information, such as similar utterances or topics. Such explicit information is sensitive to domain shift problems, since when domain shift from one to another, the explicit information may change significantly due to the topic changes in different domains. Domain shift problems influence the model performance a lot from multiple aspects, we summarize three main points below. The first two points are from the view of slots, the third point is from the view of context.

1. The first aspect is the unseen slots in new domains. Unseen slots are the slots that did not appear in source domains, so the knowledge about unseen slots were not learned in training. Although previous studies could deal with unseen slots to some extent when the unseen slots describe similar semantic topics to training slots, most unseen slots in practical scenarios are semantically dissimilar that are unrelated to training slots [86, 88, 89]. Previous methods could not work well for those unseen slots since they are difficult to be handled by the knowledge learned in the source domains. This is the most challenging problem in zero-shot slot filling and few studies [23] paid attention to this problem.

2. The second aspect is the differently explained seen slots. The differently explained seen slots are the seen slots with different explanations in new domains, such as the slot 'object name'described in the previous section. It is also difficult to treat these slots by the knowledge learned in training. Thus, this problem influences the model performance. However, this problem was not attracted much attention in previous studies.

3. The third aspect is the different context distributions in new domains. The contexts in target domains generally differ from the source domains since different domains describe different topics and scenes. This is a general problem in many research areas facing domain adaptations [15, 16, 17, 18]. The context differences have a general influence on the model performance, which degrade the accuracies of the predictions for both slot entity and slot types.

The overall objective of our research is to mine the intrinsic representations for slots and their values to address domain shift problems described above. The intrinsic representations in this research are the representations that describe intrinsic characteristics of slots and values. Generally, as the intrinsic characteristics of an object could be stable, whatever its specific appearance is. The intrinsic characteristics of slots and values could be expected to be common across domains, thus providing stable domain similarities besides explicit information. Accordingly, we expect to use intrinsic representations to provide transferable information across domains to alleviate domain shift problems.

As the intrinsic characteristics can be reflected in many aspects, such as commonsense knowledge or dense representations, we explore the representations from multiple aspects separately. Then this research utilizes the intrinsic representations in zero-shot slot filling to alleviate domain shift problems. Furthermore, this research investigates whether the improvements in zero-shot slot filling contribute to the zero-shot capacity of the task-oriented dialogue system. By alleviating domain shift problems in zero-shot slot filling and improving the dialogue system in dealing with unseen domains, our research could be expected to improve the dialogue system towards the ultimate goal of the task-oriented dialogue system.

1.3 Research Contributions

This research aims to mine intrinsic representations that describe the intrinsic characteristics of slots and their slot values to address the domain shift problems in zero-shot slot filling, and further improves the zero-shot capacity of the task-oriented dialogue system towards the ultimate goal of the task-oriented dialogue system. We introduce our contributions on general and specific levels:

On the general level, this research found that the intrinsic characteristics are effective on providing transferable information across domains for alleviating domain shift problems. Accordingly, this research gives a suggestion that exploring intrinsic characteristics within objects could benefit to domain adaptation tasks that facing domain shift problems.

On the specific level, this research explored the intrinsic representations from three aspects separately for zero-shot slot filling, including the commonsense knowledge, dense representation, and the ontology in dialogue systems:

We proposed the inference relation paths (IRPs) from the aspect of commonsense knowledge to tackle the domain shift problem 1, semantically dissimilar unseen slots. Specifically, we mined this representation from the

knowledge graph. By a statistical analysis, we found that the IRP implicitly carried the intrinsic relationships between slots and values based on the knowledge graph and was robust across domains. Then we constructed a model to utilize the IRPs in zero-shot slot filling. The experiments and analysis showed that using IRPs improved zero-shot slot filling on unseen domains by alleviating the domain shift problems, especially on semantically dissimilar unseen slots.

We proposed the multi-relation-based representation from the aspect of dense representation to deal with the domain shift problem 3, different context distribution. Specifically, we mined this representation for slot entity predictions in the two-stage approach to deal with the context corresponding to the domain shift problem 1 unseen slots and the domain shift problem 2 differently explained seen slots. Since slot entity prediction is the first stage of the two-stage approach, alleviating the domain shift problem on slot entity prediction is the premise of improving overall zero-shot slot filling. The multi-relation-based representation describes the relationships between slot entities and slot types, meanwhile considering the relationships between slot entities and various context environments. Accordingly, the multi-relation-based representation describes the intrinsic characteristic of slot entities among different slots and contexts. Therefore, the proposed representation was a high generalization capacity for slot entity predictions on a diversity of contexts in different domains. We constructed a model to utilize the multi-relation-based representations in zero-shot slot filling. The experiments and analysis showed that the proposed representation alleviated the domain shift problem in the slot entity prediction compared to previous methods.

We proposed the ontology-based representation from the aspect of the ontology in dialogue systems to deal with the domain shift problem 1 unseen slots and problem 2 differently explained seen slots. Specifically, we mined this representation for slot type predictions in the two-stage approach to deal with domain shift problems. The ontology-based representation describes the intrinsic relationships between slots and values based on the ontology knowledge base. We assume the ontology is fully-defined for both seen and unseen domains. When a domain changes from one to another, the relationships described in the ontology will not change. Therefore, the ontology-based representation is robust to the domain shift problem and effectively fills the knowledge gap of new domains in zero-shot slot filling. We constructed a model to utilize the ontology-based representation in zero-shot slot filling and showed significant improvement by alleviating the domain shift problems on the slot type prediction. We further constructed a model to utilize the multi-relation-based representation for the slot entity prediction with the ontology-based representation for the slot type prediction to compensate the

advantages of the two representations and further alleviated domain shift problems in zero-shot slot filling.

Finally, after mining intrinsic representations, this research explored whether the improvements in zero-shot slot filling contribute to the zero-shot capacity of the task-oriented dialogue system towards the ultimate goal of the task-oriented dialogue system. To do so, we employed and compared dialogue systems using different slot filling modules. The investigation results showed that alleviating domain shift problems in zero-shot slot filling generally contributes to improve the zero-shot capacity of the dialogue system when encountering unseen domains.

1.4 The organization of the dissertation

This dissertation is comprised of seven chapters. Figure 1.3 shows the organization of this dissertation. After this Chapter 1 of the introduction, the contents of Chapter 2 to Chapter 7 are briefly described as follows:

Chapter 2 reviewed the lectures on dialogue systems and zero-shot slot filling. This chapter first briefly summarized the development of dialogue systems, focusing on task-oriented dialogue systems to introduce the current stage of each module in the system. Then this chapter focused on the methods of zero-shot slot filling to give a specific background to this dissertation. After that, this chapter also introduced widely used datasets, evaluation process in zero-shot slot filling, and the current states of zero-shot slot filling.

Chapter 3 described the inference relation path (IRP). This chapter first presented the motivation and introduced what the IRP is and how the IRP could be expected to be robust across domains for zero-shot slot filling. Then, this chapter described the analysis of IRPs for various slots and their values in a large-scale dataset, the Snips dataset. The analysis showed that the IRP carries intrinsic information about slots and their values across domains for zero-shot slot filling. The next section of the chapter described the model constructions for utilizing IRPs in zero-shot slot filling. Then, this chapter described the experiments to evaluate the effectiveness of IRPs compared to previous strong baselines. The final section analyzed the results of the experiments.

Chapter 4 described the multi-relation-based representation. The first section of the chapter described the motivation for mining the multi-relation-based representation. The second section gave the representation definition and showed the advantages compared to the slot entity representations in previous studies. The following section introduced the model structure of

utilizing the multi-relation-based representation. The final section described the experiments and analyzed the results to show the effectiveness of the proposed representation.

Chapter 5 described the ontology-based representation. The first section of the chapter described the motivation for mining the ontology-based representation. The second section introduced two kinds of ontology-based representation. The following two sections described the preliminary experiments to choose a more effective ontology-based representation, including the model constructions for utilizing two kinds of representation and the experiments. After that, this chapter introduced the model structure combining the use of multi-relation-based representation and the better ontology-based representation for zero-shot slot filling. The final section described the experiments and the result analysis.

Chapter 6 described the investigation of whether the improvement of zero-shot slot filling by the proposed intrinsic representations contributes to the task-oriented dialogue system when encountering unseen domains. This chapter first introduced the motivation and the building of the task-oriented dialogue system. Then, this chapter described the experiment settings, including the experiment process, comparison methods, and evaluation metrics. The following sections described the results and analysis.

Chapter 7 summarized the dissertation and gave the conclusion of this research.

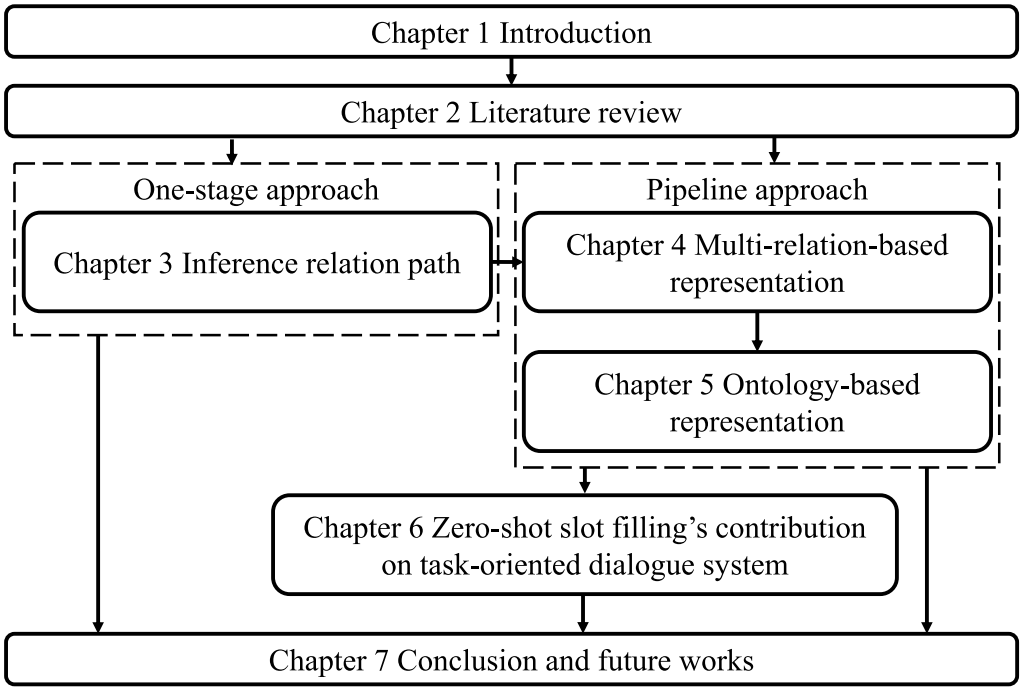

Figure 1.3: Orgnization of the dissertation

Chapter 2

Literature review

2.1 Dialogue system

A dialogue system is a computer system intended to converse with humans. Building a dialogue system that can communicate smoothly and naturally with humans is an enduring topic in artificial intelligence. In this section, we first reviewed the development of the dialogue system so far. Then, we focused on describing the present states and limitations of the widely used dialogue system, the task-oriented dialogue system.

2.1.1 The development of dialogue systems

To build a dialogue system is to make the computer understand the semantics of human languages. This has been an enduring task since the first boom of artificial intelligence from the 1960s. The first dialogue system ELIZA [1] was proposed as a medical care assistant. ELIZA is a rule-based system that converts the input utterance into a response by changing the word orders or making questions based on the input. ELIZA made dialogues with humans smoothly in the experiment, laying the foundation for dialogue systems. After that, the frame-based dialogue system [3, 4, 24], such as the GUS system [3]. GUS was proposed to help people complete tasks. GUS could understand semantic contents in the dialogue by parsing with a pre-defined set of frames representing specific semantic meanings. In the early 1990s, TOSBURG [24] was proposed as a keyword detection system in restaurant ordering scenes. TOSBURG focused on detecting the specific semantic components (keywords) to identify the customers' order and become more flexible than the early frame-based system to handle spoken languages. Then, in the early 21st century, with the development of statistical methods, the HMM-based system [5] was proposed for identifying semantic components more accurately. Due to increasing both computational powers by GPU hardware and available large-scale data, deep neural networks-based (DNN-based) dialogue systems [6, 7, 25, 26, 27] were proposed to improve the

dialogue systems for completing tasks and chatting. Such as the sequence-to-sequence dialogue generation systems [7], language understanding based on contextual encoding [25, 26], and attention mechanism [27, 28]. Recently, the pre-trained paradigm further improved the dialogue system on understanding users' requests for completing tasks and communicating with users more naturally in chatting [9, 28, 29, 30, 41]. Besides just generating texts, incorporating commonsense knowledge [31, 32, 33] and making dialogues reflecting personalized information [34, 35, 36, 37] have attracted more attention in recent years.

Nowadays, building task-oriented dialogue systems has become a hot topic in research and industry areas [10]. Our research focused on the task-oriented dialogue system.

2.1.2 Task-oriented dialogue system

The task-oriented dialogue system aims to help the user complete specific tasks by satisfying users' requests. The pipeline system [14] are popular for the task-oriented dialogue system so far, which usually consisted of multiple modules. Each module has their functions on processing specific aspect for handling dialogue, such as understanding the user's utterance or generating the response to the user. Recently, the end-to-end system were also attracted much attention [13, 108]. The end-to-end system generates responses from the input utterance directly and does not contain separate modules.

Although pipeline systems and end-to-end systems have their advantages on handling different situations [13, 14]. Nonetheless, the pipeline system has an advantage that the end-to-end system has not, that is each module of the pipeline system can be separately analyzed and studied. This makes the pipeline system more suitable on researching specific problems. The advantage of the pipeline system also makes the system more reliable in the industry area. For analyzing and solving problems that will be decribed later, our research focuses on the pipeline system.

For introducing each module in the system hereafter, we show the pipeline structure mentioned in Chapter 1 again in Figure 2.1.

From the user side, the natural language understanding (NLP) module first processes the user's input utterance by slot filling and intention detection. Slot filling identifies the intended semantic components related to the task completion within the utterance. As mentioned in Chapter 1, Slot filling is usually treated as a sequence tagging task, so the sequence modeling methods [71, 72] were general approaches for slot filling. Recently, pre-trained models were also applied for slot filling [30, 40].

The intention detection module detects the user's intention from the

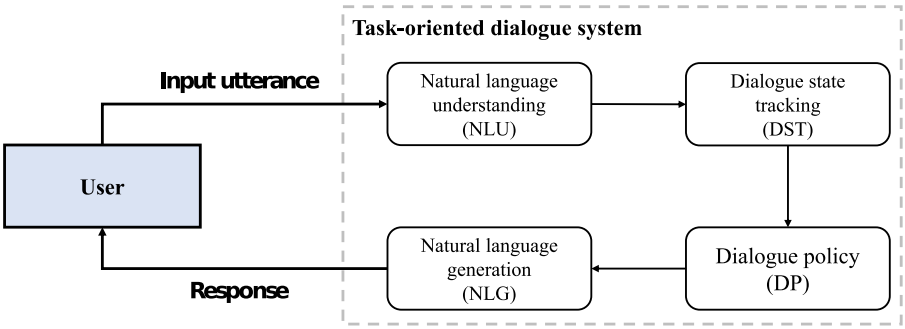

Figure 2.1: The task-oriented dialogue system's structure

input utterance. There are two definitions for the user's intentions in the task-oriented dialogue system. One describes the intended action that the user wants to convey through the utterance [42, 43, 44], such as informing information from the user itself or requesting further information from the system; The other one indicates the domain and topics the user is asking for [45, 46, 47], such as booking flights. In general, intention detection is treated as a classification task for given utterances. Accordingly, RNN-based or LSTM- based methods were popular with their advantages in encoding utterances [46, 47, 48]. Recently, similar to slot filling, pre-trained models were employed to further improve the detection accuracy [30, 49].

After NLU, the dialogue state tracking (DST) module receives the intended components and the user's intentions to update the dialogue state. Rule-based methods have been well-used for DST [14, 50, 51, 52], but this kind of method needs elaborate pre-defined rules, leading to much consumption. DST based on statistical methods was proposed to address this problem, such as DST with condition random fields [53] and maximum entropy model [54, 55]. Those methods were mainly presented for the dialogue state tracking challenge (DSTC), which is still a popular challenge for DST tasks until now. With the development of deep learning, recent works attempted to perform the process of NLU and DST jointly to obtain dialogue states from the input utterance directly [56, 57, 58, 59]. However, jointed methods were shown not to be as effective as elaborated-designed rule-based DST for the performance of the whole dialogue system [14].

After obtaining the dialogue states, the dialogue policy (DP) module select appropriate system actions for the response. Similar to DST, rule-based methods have been well-used for constructing a DP module [14, 60]. However, the rule-based DP is also limited by the pre-defined rules, so extending is not easy. Reinforcement learning-based AS was proposed [61, 62] to realize DP towards more complicated policies in the dialogue. Reinforce-

ment learning-based methods train the DP module based on dialogue states, user and system actions, and the situation of final goals. In reinforcement learning-based DP, user simulators [60, 63] are generally conducted to realize a user-system interaction without employing real users, which leads to labor consumption. However, for the overall performance of the dialogue system, the rule-based DP was shown to be effective compared to reinforcement learning approaches [14]. Besides those approaches, recent studies proposed word policies to [11, 64, 65] to jointly choose DP and generate dialogues.

Finally, the natural language generation (NLG) module generates the response to the user. The template-based NLG method [14] is a simple and effective approach to generate acceptable responses. However, similar to the rule-based approaches for other modules, the template-based NLG relies on pre-defined templates, so it is not flexible to generate responses with diverse styles. With the development of deep learning, deep learning methods were proposed to improve the semantical correctness and fluency of dialogues, such as RNN-based generation [70], semantically conditioned LSTM [67], and multi-task learning methods [68]. Recently, due to the advantages of large-scale pre-trained models, the pre-trained models have been applied for dialogue generation with fine-tuning, such as GPT-based dialogue generation models [9].

After the NLG, the dialogue system finally gives the response back to the user. With the modules introduced above, a task-oriented dialogue system could communicate with users and help users to complete specific tasks.

2.1.3 Zero-shot methods for building domain adaptable task-oriented dialogue system

With the development of applications and services, an existing dialogue system is required to handle more domains describing different topics and requests.

In the extension of the dialogue system to different domains, the domain shift problem is the main problem since different domains have diverse contexts and topics. A system usually hardly deals well with such differences since the knowledge of new domains was not learned in training. Retraining models to adapt to new domains is a typical approach to alleviate the domain shift problem. However, as the requirements for domain extension increase, retraining the system often is generally inefficient and time-consuming in real-world business.

For the problem above, one of the ultimate goals of the task-oriented dialogue system was proposed [10]. This ultimate goal is to build a system

that is adaptable to any given domain without any training instance of the given domain, so-called zero-shot domain adaptation. The system that meets this goal is expected to handle any extensions of domains in practice. Meanwhile, no retraining is required.

Towards this ultimate goal, conventional methods for modeling the modules in the system are insufficient. The main reason is that conventional methods are hard to handle this situation: when new domains are encountered, not only the new distribution of semantic information encountered, but new classes that need to be predicted may also be encountered. Since these new classes were unseen in the training process, conventional methods are unable to deal with those classes as there is no output to obtain the predictions for the unseen classes.

Transfer learning methods [17] were general approaches to address the domain shift problems for domain adaptation. Towards the ultimate goal of the task-oriented dialogue system, the zero-shot paradigm from transfer learning is suitable for handling unseen classes in new domains. Zero-shot methods were originally proposed in the computer vision area [73, 74] and extended to the natural language understanding area [75, 76]. The basic idea for realizing zero-shot methods was to utilize the similarities between the classes in source domains and the zero-shot classes in target domains [74]. Recent methods realized zero-shot methods by designing training and prediction strategies to avoid retraining [19, 21]. These methods train the model on source domains and directly adapt the model on zero-shot target domains. The domain adaptations are relied on the similarities between domains rather than between classes. The similarities between domains can be reflected on many aspects, such as utterances, vocabularies, and classes. The following reviewed lectures are mainly based on the latter paradigm.

In the task-oriented dialogue system, since the performance of each module influence the whole system's performance, each module's zero-shot capacity is related to the system's performance in dealing with domain shift problems. Thus, previous studies for zero-shot methods on each module have been conducted in recent years.

For the slot filling module in NLU, zero-shot slot filling [19, 22] was proposed to deal with unseen slot classes by predicting slot entities and slot types separately. Detailed reviews of zero-shot slot filling will be introduced in the next section.

For the intention detection module in NLU, previous studies were conducted based on the intention definitions described in Section 2.1.1. For the intention that indicates the domain and topics the user is asking for, the intention is basically equivalent to the domain. Accordingly, the intention changes when the system is extended to new domains. Therefore, the zero-

shot intention detection for such an intention definition is usually handling zero-shot domain detection. Zero-shot domain detection aims to identify the domain of the user's utterance. Previous studies used the capsule network to obtain the probabilities of the unseen domains considering the relationships between classes and context situations [77]. The capsule network is a structure that could describe classes considering not only the representation of the classes themselves but also multiple components [78]. Further works using the capsule network improved the zero-shot domain detection by not only identifying the correct unseen domains but also detecting the correct domain from both training domains and unseen domains [79]. For the intention that describes the user's intended action, the intention is generally stable across domains since the actions of users and systems are generally described in a limited and small set [42, 43]. Therefore, zero-shot intention detection for such an intention definition was not attracted much attention.

For the DST module, the dialogue states are dynamic in different domains. Previous studies handled dynamic dialogue states by merging NLU and DST processes and used similarity-based methods to obtain dialogue states directly from the given utterance [12, 59, 91]. The combining of NLU and DST processes means that previous methods need to handle not only the unseen dialogue states in new domains but also face the domain shift problem caused by the unseen slots. Therefore, the zero-shot setting for DST is more demanding, so the accuracy was limited.

For the DP and NLG modules, although they face domain shit problems when the dialogue system encounters new domains, zero-shot settings are not urgently required. This is because the policies can be described in a universal and finite set across different domains, so the conventional DP approach still works in the zero-shot scenario. While the NLG module mainly faces out-of-vocabulary (OOV) words and dialogue style changes. The OOV problem has been addressed much in the dialogue generation research [80, 81, 82]. The dialogue style changes can be alleviated by pre-defined templates, so the conventional NLG approaches could also work in zero-shot scenarios.

According to previous studies introduced above, towards the ultimate goal of the task-oriented dialogue system introduced above, the slot filling module and the DST module are more required to be able to handle zero-shot scenarios. Based on the result in previous studies [14] that a complete pipeline task-oriented dialogue system performed significantly better than a dialogue system with a merged NLU + DST module, and with the consideration that the outputs of slot filling are the basis of subsequential process, the zero-shot capacity of slot filling is considered to determine the system's zero-shot capacity to a large extent. Therefore, our research concentrates on zero-shot slot filling, focusing on alleviating domain shift

problems towards the ultimate goal of the task-oriented dialogue system.

2.2 Zero-shot slot filling

Zero-shot slot filling is to train a model on source domains and adapt the model to target domains directly without any training instances. Specifically, the model can only use the instances and slot sets in source domains in the training process. The model cannot use any instance from target domains in the training process, no matter whether the instances belong to the unseen slots that only appear in target domains or the seen slots that appear in source and target domains. In the test process, the model performs zero-shot slot filling based on the premise that the slot sets of target domains are known. Moreover, the model can use the knowledge of the slots in target domains, such as the values of slots used in previous work [21]. As mentioned in the previous section, zero-shot slot filling methods were designed to correlate source domains and target domains rather than directly correlating classes. Specifically, zero-shot slot filling methods use implicit similarities between source and target domains to provide transferable information to handle zero-shot domain adaptation. The implicit similarities can be reflected in multiple aspects, including slot types and context situations.

To realize zero-shot slot filling, two popular approaches were proposed: the one-stage method [19, 20, 21] and the two-stage pipeline method [22, 23, 83]. Recently, with the development of pre-trained models, pre-trained-based methods were also proposed [84, 85]. However, the pre-trained models are in different paradigms with typical zero-shot slot filling approaches because they may have learned knowledge about new domains and slots in the pre-training process. Our research aims to explore intrinsic representations to alleviate domain shift problems under the typical setting of zero-shot slot filling described above, no knowledge of new domains can be learned priorly. Moreover, if the representation in the typical setting is effective, it could be expected to be effective for improving pre-trained models. Accordingly, the pre-trained paradigm is an aside approach in our research.

2.2.1 One-stage approach

The one-stage approach was first proposed to handle unseen slot classes for zero-shot slot filling [19]. One-stage methods aim to identify the slot entities for each slot type in the given domain by considering the context information and slot type information. The context information can be obtained by encoding the utterance. The slot type information is usually

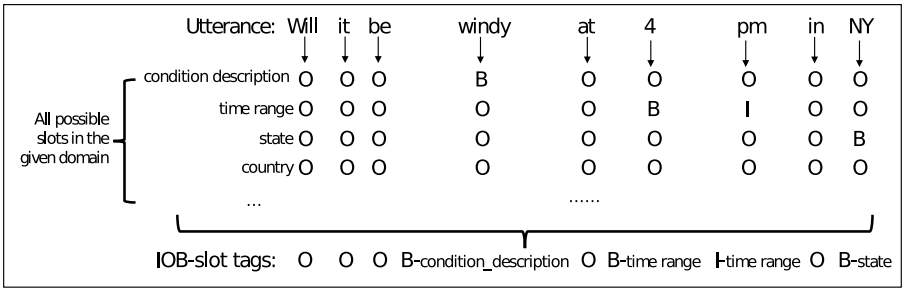

Figure 2.2: Basic process of the one-stage approach

obtained from the representation of slot descriptions. Here a slot description indicates the text description of a slot. The slot name is generally used as the slot description [19, 21]. While the sentence description was also used in previous studies [20] but the sentence description is not universal for different datasets and different human-defining.

Figure 2.2 explains the basic idea of the one-stage methods. As seen in the figure, for the given utterance 'Will it be windy at 4 pm in NY', an one-stage model identifies corresponding slot entities for each slot type in the given domain from the utterance. For instance, the model identifies 'windy' for the slot 'condition description,' '4 pm' for the slot 'time range,' and 'NY' for the slot 'state.' While the model does not predict slot entities for other possible slots in the domain. Finally, the model merges the predictions for all possible slot types and outputs the results. In this process, an one-stage model only classifies the IOB tags to indicate the chunk of slot entities, so the model could still predict the slot entity chunks for the unseen slots without modifying the output layer for IOB tags even if unseen slots were encountered.

Figure 2.3 shows a representative one-stage method, the concept tagger (CT) [19]. In the model process, an encoder first encodes the input utterance into context representation. Then the model concatenates the given slot's representation with each token's context representation. Finally, another encoder with the IOB classification layer is used to encode the combined representation to obtain the probability for IOB tags. The IOB tags indicate the slot entity chunk for the given slot type. After predicting IOB tags for each given slot type, the model merges the slot type tags with corresponding IOB tags into an IOB-slot tag sequence as the final output. CT had a simple structure and realized zero-shot slot filling for unseen slots.

The zero-shot adaptive transfer network (ZAT) [20] was proposed to improve the generalization capacity of the CT. ZAT employed an attention mechanism to obtain the context-aware slot representation to improve the

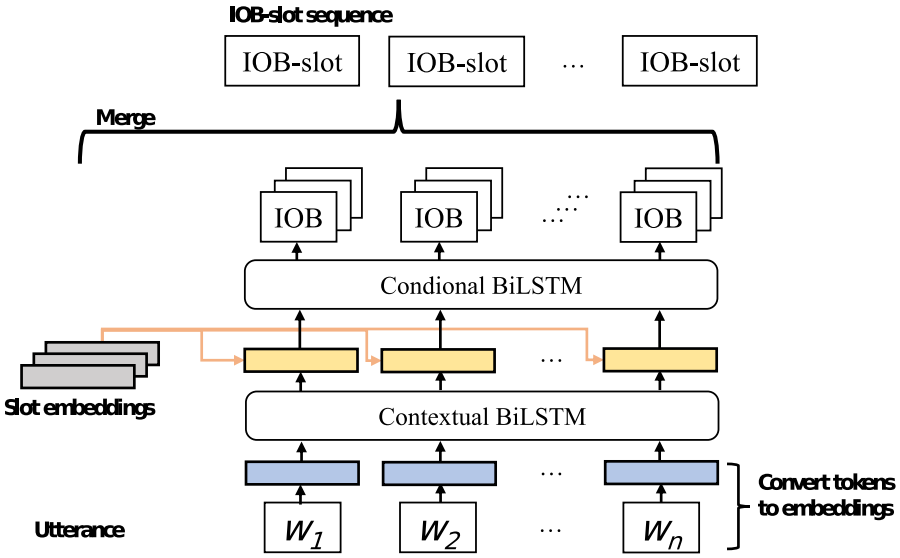

Figure 2.3: The structure of CT

generalization capacity for dealing with different context situations. ZAT also added a CRF layer for the IOB classification to improve the sequence modeling in predicting slot entity chunks.

To improve the model's robustness in handling the slots in new domains that are semantically similar to the training slots, the robust zero-shot transfer learning with examples (RZT) [21] was proposed using slot value examples. RZT added several slot values as example values with the assumption that semantically similar slots share similar slot values. For instance, the slot 'leave by' is a slot in the training domain of booking trains, while the slot 'depart at' is an unseen slot in the unseen domain of booking flights. The slots 'leave by' and 'depart at' are semantically similar, and they both represent the meaning of leaving time. 'leave by' may have slot values such as '12:00' and '13:00,' while 'depart at' may have slot values such as '14:00' and '15:00.' Accordingly, the slot values are semantically similar. RZT used slot values that could carry additional information for dealing with unseen and semantically similar slots and improved the robustness of zero-shot slot filling on unseen domains.

In summary, one-stage methods were proposed to handle the predictions for unseen slots and realized zero-shot slot filling. The developments of one-stage methods improved the generalization capacity and the accuracy towards specific slots (i.e., semantically similar slots in the new domains). However, one-stage methods were argued to be less capable of predicting

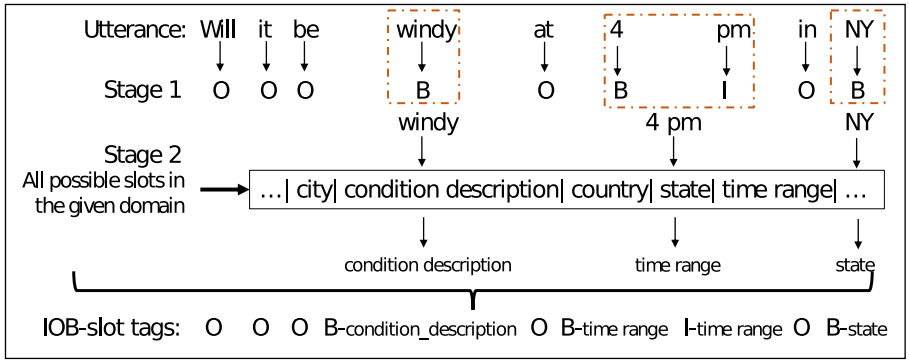

Figure 2.4: Basic process of the two-stage approach

whole slot entities [22] for new domains. This is because one-stage methods learn to predict slot entities considering the context situations for each specific slot type, but the context environments in new domains could be much different from that in training domains due to domain shift problems. Moreover, in the prediction process of one-stage methods, the I and B tags are separately predicted from utterances for one given slot type. This lead to low interpretability of explaining the predictions for slot entities of the slot type.

2.2.2 Two-stage approach

To address the problem that one-stage methods are less capable of predicting whole slot entities, the two-stage pipeline approach was proposed [22, 23, 83] to predict slot entities and slot types separately. Figure 2.4 explains the basic idea of the two-stage approach. As seen in the figure, in stage one, a model first identifies general slot entities from the given utterance without considering specific slots. For instance, 'windy,' '4 pm', and 'NY' are identified as slot entities in stage one. In stage two, the model uses the algorithm without learning parameters (e.g., similarity matrix) to obtain the slot type predictions for each slot entity. For instance, the two-stage model predicts 'windy' belonging to the slot 'condition description,' the '4 pm' belonging to the slot 'time range,' and the 'NY' belonging to the slot 'state.' Finally, the model merges slot entity predictions from stage one and slot type predictions from stage two into an IOB-slot type sequence as the final output.

Figure 2.5 shows a representative two-stage method, coarse-to-fine approach (Coach) [22]. In stage one, Coach used a sequence encoder-decoder to predict IOB tags without considering the specific slot information like

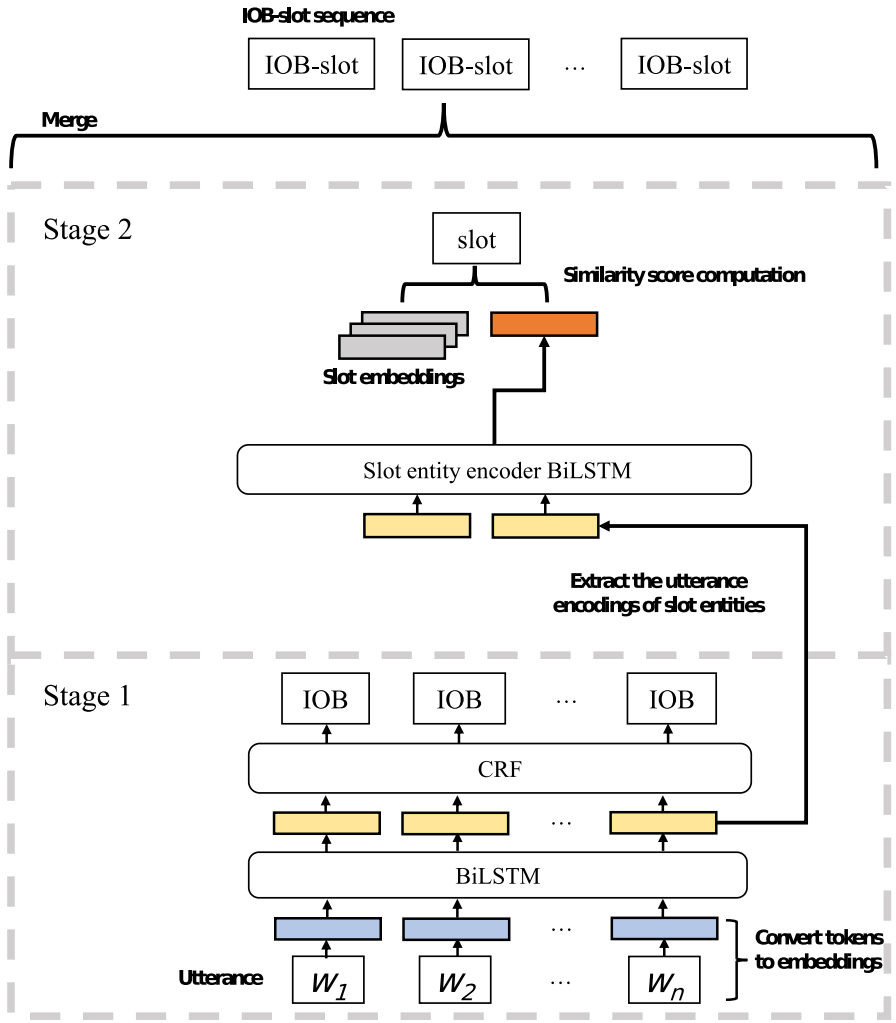

Figure 2.5: The structure of Coach

one-stage methods do. Therefore, Coach learns general slot entity patterns among various context environments to improve the generalization capacity of slot entity predictions. In stage two, Coach extracted the slot entities indicated by the IOB tags and used another encoder to obtain slot entity encoding representations. Next, Coach concatenates the representations of all possible slot types' slot descriptions in the given domain into a matrix. This matrix is then multiplied with the slot entity encoding representation to obtain similarity scores between the slot entity and each slot type. The slot type with the highest score was chosen as the prediction. Finally, Coach merged the predictions from stage one and stage two as the final output.

After Coach, contrastive zero-shot learning with adversarial attack

(CZSL-adv) [83] was proposed to employ contrastive learning to improve the learning for domain information. Prototypical contrastive learning and label confusion (PCLC) [23] pointed out that the improvement on unseen slots was limited and used prototype learning and label confusion to improve the prediction for unseen slots.

In summary, the two-stage methods improved zero-shot slot filling than one-stage methods [23], especially on unseen slots. The two-stage structure made the zero-shot slot filling more interpretable than one-stage methods because separate predictions of slot entities and slot types allow the model to correlate slot types with complete slot entities rather than I or B tags. The two-stage method has the potential to be developed by improving each stage to improve zero-shot slot filling benefiting from the separate prediction.

2.2.3 The datasets for zero-shot slot filling

For evaluating zero-shot slot filling, previous studies conducted experiments on multi-domain datasets.

The most widely used dataset to evaluate zero-shot slot filling is the Snips dataset [86]. The Snips dataset is a multi-domain dataset containing seven domains. These domains are all describing the user's requests for smart devices. Each domain contains about 2000 utterances of user speakers alone. There are no system speakers in the Snips dataset. So far, two versions of the Snips dataset are available, one version was used previous studies [22, 23], and the other version was not. The two versions has the same domains and slot sets. The number of utterances in each domain were similar, which were about 2000 utterances. The difference was on the contents of the utterances. We compared two versions and found that averagely 14% of utterances are different among seven domains between the two versions. To distinguish the versions, we call the version used in previous studies as the 'Snips dataset' hereafter. And we call the other version as the 'Snips2017 dataset' since it was benchmarked in 2017. In our research, we used the Snips2017 dataset in Chapter 3. Then, we used the Snips dataset in Chapter 4 and Chapter 5 to fairly compare with baselines, as the baselines in these two chapters used the Snips dataset in their official implementations.

The MultiWOZ 2.0 dataset [42] is another large-scale multi-domain dataset for task-oriented dialogue system tasks. This dataset contains seven domains that describe the tasks and scenarios in daily life, such as booking hotels and trains. The dataset contains dialogues of user-system interactions, the utterances from both the user and system speaker were annotated with slots and dialogue actions. The MultiWOZ 2.0 dataset has been developed in several versions. The MultiWOZ 2.1 [87] and MultiWOZ 2.2 [107] corrected

annotation errors in the original dataset. The latest version is MultiWOZ 2.3 [88], which added the annotations of coreference resolution to adapt the dataset to more tasks. We use the MultiWOZ 2.3 dataset in this research.

The Schema Guided Dialogue (SGD) dataset [89] is an unprecedented large-scale dataset for diverse tasks in the task-oriented dialogue system. The SGD dataset contains 20 domains describing various scenarios of user-system interactions, such as setting alarms, booking hotels, and asking for the weather. Due to the scale of the SGD dataset, previous studies were extended for dialogue tasks on a large scale [90]. Similar to the MultiWOZ dataset, the SGD dataset contains dialogues from the user and the system speakers. The utterances are annotated with slots and dialogue actions, and the dialogues are annotated with dialogue states. We use the current version of the SGD in this research.

For each dataset, the slots mentioned in each domain can be categorized by how they appear in different domains: multi-domain slots are shared by multiple domains, and single-domain slots are contained by only one domain. For instance, in the Snips dataset, the slot 'city' mentioned in the Book restaurant domain and the Get weather domain is an multi-domain slot; while the slot 'served dish' only mentioned in the Book restaurant domain is a single-domain slot.

2.2.4 Evaluation process and metrics

In evaluating zero-shot slot filling, previous studies mainly used the left-one strategy [19, 20, 21, 22, 23]. Specifically, for a given dataset containing n domains, one domain is set to be the zero-shot target domain, and remaining n-1 domains were used as source domains. Then, n separate models were trained by setting each domain as the zero-shot target domain. The evaluations were mainly on the performance of each target domain and the average performance among all models. We also follow this evaluation process in this research for fair comparisons with previous studies. Accordingly, the single-domain slots are unseen slots if the corresponding domain is set to be the zero-shot domain since those slots appear in the zero-shot domain alone and would not be seen in training domains. The multi-domain slots are seen slots if one of the corresponding domains is set to be the zero-shot domain since those slots appear in training domains.

The slot F1 score [92] is the most popular metric to evaluate the performance of zero-shot slot filling. The slot F1 score computes the F1 score from chunks of each slot. A chunk is a complete slot entity corresponding to a specific slot type. For instance, 'windy,' '4 pm,' and 'NY' in Figure 2.2 are chunks corresponding to their slots.

In the process of slot F1 score computation, true positive (TP), false positive (FP), true negative (TN), and false negative (FN) samples are first counted for each slot's chunks. For a given slot s_i, TP_i counts the chunks that should be predicted to s_i were predicted correctly; FP_i counts the chunks that should not be predicted to s_i were predicted as s_i; TN_i counts the chunks that should not be predicted to s_i were predicted not belong to s_i correctly; FN_i counts the chunks that should be predicted to s_i were predicted not belong to s_i.

Then, the slot F1 score can be used to compute the F1 score for overall performance and each slot's performance. For the overall performance, the slot F1 score can be computed as:

$$precision = \frac{\sum_{i=1}^{N_s} TP_i}{\sum_{i=1}^{N_s} TP_i + FP_+ i} \tag{2.1}$$

$$recall = \frac{\sum_{i=1}^{N_s} TP_i}{\sum_{i=1}^{N_s} TP_i + FN_+ i} \tag{2.2}$$

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{2.3}$$

where N_s is the set of all slots in the slot filling results, *precision* and *recall* are the precision and recall computed for all slots, respectively.

For the performance of a given slot s_i, the slot F1 score $F1_i$ can be computed as follows:

$$precision_i = \frac{TP_i}{TP_i + FP_i} \tag{2.4}$$

$$recall_i = \frac{TP_i}{TP_i + FN_i} \tag{2.5}$$

$$F1_i = 2 * \frac{precision_i * recall_i}{precision_i + recall_i} \tag{2.6}$$

where $precision_i$ and $recall_i$ are the precision and recall for slot s_i, respectively.

Besides the slot F1 score on evaluating the performance of slot filling, the IOB F1 score can be used to evaluate the performance of the slot entity predictions alone. IOB F1 score computes the F1 score for chunks alone without considering specific slot types. Similar to the process of computing slot F1 score, TP_{IOB}, FP_{IOB}, TN_{IOB}, FN_{IOB} for the IOB F1 score are first counted to compute the IOB F1 score. TP_{IOB} counts the chunks that should be a complete slot entity were correctly predicted; FP_{IOB} counts the chunks

that should not be a complete slot entity were predicted as a slot entity; TN_{IOB} counts the chunks that should not be a complete slot entity were predicted not to be a slot entity; FN_{IOB} counts the chunks that should be a slot entity were predicted not to be a slot entity. Then the IOB F1 score $F1_{IOB}$ could be computed as:

$$precision_{IOB} = \frac{TP_{IOB}}{TP_{IOB} + FP_{IOB}} \tag{2.7}$$

$$recall_{IOB} = \frac{TP_{IOB}}{TP_{IOB} + FN_{IOB}} \tag{2.8}$$

$$F1_{IOB} = 2 * \frac{precision_{IOB} * recall_{IOB}}{precision_{IOB} + recall_{IOB}} \tag{2.9}$$

where $precision_{IOB}$ and $recall_{IOB}$ are the precision and recall computed for slot entities, respectively.

2.2.5 The current states of zero-shot slot filling performance

Next, we introduce the current states of zero-shot slot filling performances based on evaluation process. Table 2.1 lists a domain averaged results on the Snips dataset from the paper of Coach [22] and PCLC [23]. The results were from the one-stage methods of CT and RZT, and two-stage methods of Coach, and PCLC. As seen in the table, two-stage methods Coach and PCLC generally outperformed one-stage methods CT and RZT, and PCLC was reported to achieve the state-of-the-art performance.

However, from the official implementations of PCLC and Coach, we found these methods have two premises on achieving the good performance. One is that they used modified slot names as slot descriptions, which is not universal descriptions for slots in different datasets. For instance, PCLC and Coach used 'owner' as the description for the slot type 'playlist owner' in the Snips dataset, which changed the semantic meaning of the original slot. In practice, the exact slot name should be used for describing the slot to guarantee that the slot description is universal and available in different domains. For the previous instance, the description of the slot 'playlist owner' should be the words of 'playlist owner.'

The other premise is due to the version of the neural network framework, PyTorch. The official implementations was based on old versions of PyTorch, which were v1.3 and v1.4. Those versions allowed PCLC and Coach used an incorrect optimization process that updated the model parameters multiple

Table 2.1: The performance of zero-shot slot filling reported in previous studies

Dataset	CT	RZT	Coach	PCLC
Snips	30.55	32.85	37.39	42.82

Table 2.2: The performance of zero-shot slot filling with universal slot descriptions and correct optimization processes

Dataset	CT	ZAT	RZT	Coach	PCLC
Snips	37.94	38.06	**38.50**	33.42	30.93
Snips2017	33.50	**40.72**	33.22	40.03	28.20
MultiWOZ 2.3	**65.91**	-	58.51	59.55	57.01
SGD	57.80	-	56.03	**68.72**	63.32

times based on the loss computed from the non-updated model parameters. Specifically, in the training process, these two-stage methods obtained the slot entity predictions' distributions and slot type predictions' distributions for each slot entity. Then they computed the loss of slot entity predictions and performed loss backward to update the model parameters of stage one. After that, they used the distribution of pre-obtained slot type predictions to compute the slot type predictions' loss and updated model parameters of stage one and stage two. However, in this updating process, stage one's parameters should not be updated again by the slot type predictions' loss since the pre-obtained slot type predictions were not obtained from the updated parameters of stage one. Moreover, the loss of slot type predictions should only correspond to stage two since the loss was computed by the slot type predictions' distributions alone. To correct the optimization process, the slot type predictions' distributions should be detached from stage one at first. Then the loss could be computed for stage two to update model parameters.

To reduce the influence of the premises and to clarify the current states of zero-shot slot filling, we reproduced PCLC, Coach, CT, and RZT based on the official implementations and our own implementations. The evaluation setting and metrics are based on the process described in the previous subsection. We reproduced previous studies not only on the Snips datasets, but also on Snips2017, MultiWOZ 2.3, and SGD datasets to make a comprehensive clarification. Moreover, we reproduced ZAT on Snips and Snips2017 datasets although previous studies did not compared with ZAT. We did not reproduced ZAT on MultiWOZ 2.3 and SGD datasets since ZAT have an unacceptable time-consuming on the two datasets. This inefficiency

Table 2.3: The best method on each dataset

Dataset	Best Method
Snips	RZT
Snips2017	ZAT
MultiWOZ 2.3	CT
SGD	Coach

problem of ZAT can be considered to be the reason that previous studies did not compared with it.

Table 2.2 lists the domain average performance of different methods on each dataset. The bold numbers indicate the best results on each dataset. As seen in the table, the PCLC and Coach degraded their performance with universal slot descriptions and corrected optimization processes on the Snips dataset. In contrast, one-stage methods CT and RZT showed better performances than two-stage methods on the Snips dataset. By comparing the performances on each dataset, we found that no one method had a leading performance on all dataset, each model had their advantages on different datasets. Accordingly, the two-stage methods are not always better than one-stage methods as shown in previous studies, constructing a zero-shot slot filling model should choose the framework according to the situations.

Based on the results in Table 2.2, we summarize the best method for each dataset in Table 2.3. With the clarifications on the best methods for each dataset, we can evaluate whether our proposed methods generally well-performed or not in subsequential chapters by comparing with the best methods .

2.3 Problems in zero-shot slot filling

The ideal performance of zero-shot slot filling on unseen domains should be as good as conventional methods trained on those domains. However, the performance of zero-shot slot filling was limited and still not comparable to conventional slot filling methods on specific domains. For instance, the best zero-shot method RZT achieved 38.50 of slot F1 score on the Snips dataset, while a well-performed conventional method BERT achieved about 97.00% of slot F1 score on the Snips dataset [49]. As described in Chapter 1, the main reason is that the previous studies mainly relied on the similarities based on explicit information, such as similar topics and contexts. Such information is sensitive to domain shift problems since when the domain changes to new domains, the text and slots also change to make the explicit information much

different from training domains. The intrinsic characteristics of slots and values can be expected to provide transferable information besides explicit information. Although some representations in previous studies [22] actually captured the intrinsic characteristic of values, they did not focus on utilizing intrinsic characteristics on alleviating domain shift problems much so the improvements were not high. In this research, we focus on mining intrinsic representations that describe the intrinsic characteristics of slots and values to provide effective transferable information across domains to alleviate domain shift problems of unseen slots, differently seen slots, and different context distributions.

Chapter 3

Inference relation path

3.1 Objective

As described in previous chapters, previous studies on zero-shot slot filling mainly relied on explicit information and suffered from domain shift problems. In the three domain shift problems summarized in Chapter 1, the problem of unseen slots is the most challenging problem. If the descriptions of unseen slots are semantically similar to that of some training slots, previous methods could work. However, due to the domain shift problem, previous methods were hard to handle the unseen slots that are not semantically similar to training slots. In this chapter, we propose the inference relation path that describe the intrinsic relationships between slots and their values via the knowledge graph to address the domain shift problem of semantically dissimilar unseen slots.

3.2 Inference relation path

For the problem that conventional slot descriptions were ineffective in providing transferable information for handling semantically dissimilar slots in new domains, we move our sights to how people estimate objects with less semantic similarity. In practice, people often use many implicit semantic relations as estimation cues, such as inferring the situation of the fish from the kind of fishing rod the professional is using. The rob is an implicit semantic relation carrier, implying the information about the fish's situation. Inspired by this, we focus on finding a semantic relation carrier that can provide the semantic relation information between slots and their values. We try to utilize such a semantic relation carrier to estimate unseen slots' values in zero-shot slot filling. Since human inferences are often utilized for obtaining semantic relations between the intent and target, analogically, the inference paths between slots and values may have implicit semantic relations. The knowledge of entities and their relations is necessary to describe the inference between slots and values. Therefore, we use the knowledge graph

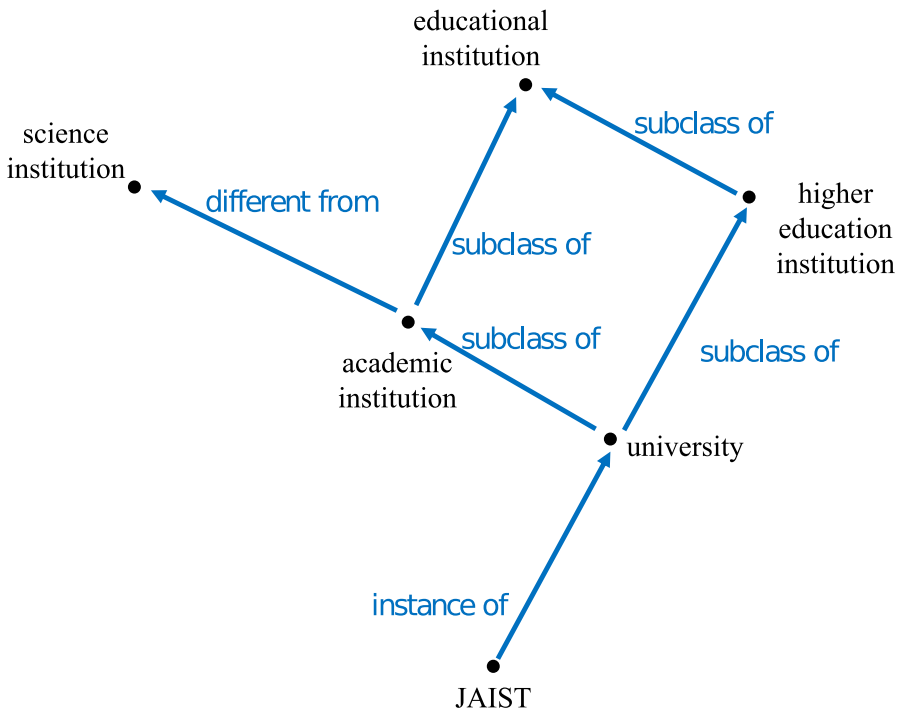

Figure 3.1: Example of nodes and edges in the knowledge graph

that describes the entity knowledge and the relations of entities to mine intrinsic representations of slots.

In the knowledge graph, entities are described as nodes, and the edges connecting entity nodes represent the relations between entities. For example, Figure 3.1 illustrates a part of the knowledge graph Wikidata to show the entity nodes and relation edges. As seen in the figure, 'JAIST', 'university', and 'academic institution' are the entities described as nodes in the knowledge graph. 'Instance of,' 'subclass of,' and 'different from' are the relations between entities. The relations are described as edges in the knowledge graph. We can infer from one entity node to another by a specific path of relation edges. The entities and relations within an inferring process consist of an inference path. For instance, we can infer from the entity 'JAIST' to the entity 'educational institution' with the inference path of [JAIST, instance of, university, subclass of, higher education institution, subclass of, educational institution].

In our research, we aim to find a semantic relation carrier that can correlate the semantically dissimilar slots. For this purpose, we focused on relations rather than entities since entities were usually used for providing

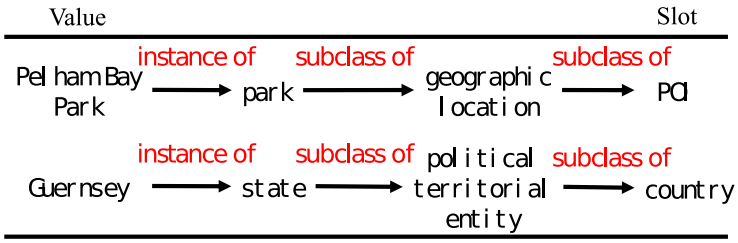

Figure 3.2: Examples of inference paths from values to corresponding slots in the Snips2017 dataset

explicit semantic information [93, 94]. Such explicit information hardly provides transferable information to handle semantically dissimilar unseen slots. On the other hand, the relations are in a finite set pre-defined in the knowledge graph. We speculated that the relations in different inference paths could be similar to some extent.

To preliminarily verify the speculation, we first tried to extract several inference paths between slots and their values from the Snips2017 dataset via a large-scale knowledge graph Wikidata. Since the Snips2017 dataset does not contain the knowledge base describing the slots and all their values (i.e., ontology), we first extracted several slots and corresponding values for each slot based on annotated utterances. After that, we extracted the inference path from the value to the slot via Wikidata. To do so, we first identified the entity nodes of the value and the slot via Wikidata, respectively. Next, we performed the route search from both the value node and the slot node. Then, we obtain the shortest inference path from the value node to the slot node by finding the intersection of the inference paths that began from the two nodes. We set such a shortest inference path as the inference path from the value to the slot.

Figure 3.2 shows two examples of the inference paths from values to the corresponding slots. One is from the value 'Pelham Bay Park' to the slot 'POI.' The inference path is [Pelham Bay Park, instance of, park, subclass of, geographic location, subclass of, POI]. The other one is from the value 'Guernsey' to the slot 'country.' The inference path is [Guernsey, instance of, state, subclass of, political territorial entity, subclass of, country]. As seen in this example, the inference paths are different. However, if we focus on the relations alone, the inference relation paths (IRPs) are [instance of, subclass of, subclass of] and [instance of, subclass of, subclass of], which are the same. Therefore, although the entity nodes in different inference paths are different, the IRPs within the inference paths could be similar. Moreover, the values 'Pelham Bay Park' and 'Guernsey' represent place names. Accordingly,

a certain IRP may implicitly correspond to certain semantic information, correlating slots to certain values.

3.3 Statistical analysis

To testify to the generality of an IRP carrying certain semantic information implicitly, we analyzed a number of IRPs between the slots and their values in the Snips2017 dataset. The Snips2017 dataset contains seven domains and 53 slots in total (We treat same-named slots in different domains as different slots). We extracted all slots and their values from the annotated utterances to build an ontology that contains all slots and their values. Since the numbers of each slot's values are much different (from one to over hundreds), we then randomly selected up to 30 values for each slot to balance the values of each slot for extracting IRPs. As a result, we obtained a knowledge base containing all 53 slots, each with up to 30 values.

Then we identified the entity nodes of the 53 slots and their values via Wikidata to obtain the IRPs between slots and their values. In the identification process, some slots could not be identified as entity nodes directly. We explored the entity in the knowledge graph with similar semantic meanings to the slots and then used the new-found entities as such slots' entity nodes to extract IRPs in subsequent processing. For instance, the slot 'served dish' could not be identified as an entity node directly. We found that the entity 'dish' describes a similar semantic meaning to the slot 'served dish,' so we use the entity 'dish' for extracting IRPs for the slot 'served dish.'

Next, depending on whether the slot's values could be identified as entity nodes or not, we separated the IRPs into two groups: (1) If at least one value of a slot could be identified as entity nodes in Wikidata, we extract the IRPs by the process described in the previous section; (2) For some slots, no corresponding values could be identified as entity nodes in Wikidata. For instance, the values of the slot 'playlist owner' are all possessive pronouns that could not be identified as entity nodes. We set the IRP as [instance of] between such slots and their values. After this processing, the slots in Group (1) had multiple kinds of IRPs corresponding to their values., and the slots in Group (2) had only one kind of IRP, which is [instance of]. Among the 53 slots in the Snips2017 dataset, 46 slots belonged to Group (1), and seven slots belonged to Group (2).

For analyzing the extracted IRPs, we converted the values corresponding to each IRP into vector representations via word embeddings. Then we clarified the distributions of the IRP-corresponded values in the semantic space. Figure 3.3 shows an example of IRP-corresponded value distributions.

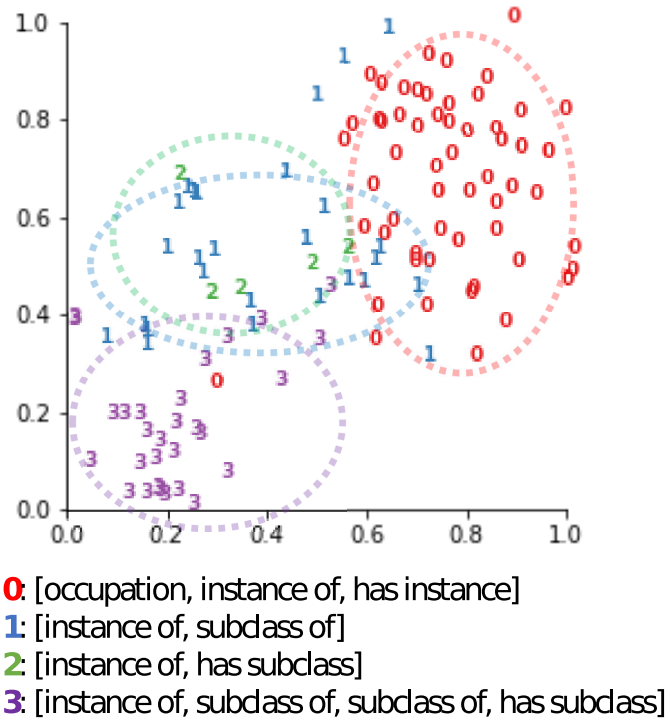

Figure 3.3: Visualization example of value distributions in semantic space corresponding to four IRPs

To explain the distributions intuitively, we reduce their dimensions using the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm [95] to obtain the distributions in the 2-D semantic space. To explain the value distributions in the semantic space clearly, we selected four representative IRPs, referred to as IRP 0 to IRP 3. Specifically, IRP 0 is [occupation, instance of, has instance]; path 1 is [instance of, subclass of]; path 2 is [instance of, has subclass]; path 3 is [instance of, subclass of, subclass of, has subclass]. Accordingly, IRP 1 and IRP 2 are similar since they have only one different step, while other steps are identical. IRP 0 and IRP 3 differ from each other and are also different from IRP 1 & IRP 2. Therefore, those IRPs could be treated as three groups: IRP 0, IRP 1 & IRP 2, and IRP 3. Figure 3.3 illustrates the value distribution corresponding to the four IRPs. As seen in the figure, the values corresponding to IRP 0, IRP 3, and IRP 1 & 2 are distributed separately as three clusters in the semantic space, where the values corresponding to IRP 1 and IRP 2 are overlapped heavily. The analysis of IRP-corresponded value distribution confirmed that the IRPs carry implicit semantic information, correlating the slot and its values

with certain semantic meanings. Therefore, the IRPs could be regarded as semantic cues that implicitly carry semantic relation information of slots and their values. This kind of relation information describes the intrinsic characteristics of slots and the relationships between slots and their values and are rarely related to domains and contexts. If we treat the IRPs as a property of a slot, similar to estimating the fish's situation by the rob that the professional fisher is using, we can estimate the values based on the slot's IRPs. Therefore, the IRP can provide transferable information for estimating the values of unseen slots.

Furthermore, we analyzed the IRPs and found that the IRPs of each slot can be grouped into several patterns for efficient use in practice. For instance, the slot 'artist' has 30 corresponding values due to the process above. Among them, the IRPs from 28 values to the slot are the same, which is [occupation, instance of, has instance]. The IRPs from the other two values to the slot are in another pattern, which is [instance of, subclass of, subclass of, has part]. Accordingly, the IRPs of the slot 'artist' could be grouped into two IRP patterns.

Next, we investigated whether or not similar IRP patterns exist in different domains to confirm whether the IRP could be used for providing transferable information in practice. For this purpose, we count the number of the slots' IRP patterns in one domain that are similar to those in other domains. Table 3.1 shows the counted results. Note that the numbers in the table do not reflect the amount of transferable information provided by IRP patterns since the effectiveness of transferable information is task-dependent. In Table 3.1, SD and MD are the number of counted IRPs of the single-domain and multi-domain slots, respectively. As seen in the table, numerous similar IRP patterns exist in different domains for SD slots and MD slots. Therefore, IRPs could be used to provide transferable information on tackling the domain shift problem for unseen slots in practice.

Table 3.1: The same IRP patterns among slots in all domains in the Snips2017 dataset

Domains	SD slot	MD slot
AddToPlaylist	5	7
BookRestaurant	28	17
GetWeather	8	10
PlayMusic	19	11
RateBook	4	7
SearchCreativeWork	0	14
SearchScreeningEvent	5	7

3.4 Experiments

3.4.1 Graph embeddings

Before introducing the model structure, to utilize IRPs in zero-shot slot filling, we first introduce the graph embedding representations that are necessary to convert IRPs into vector representations. Graph embedding is a kind of algorithm to convert the nodes and edges into vectors while guaranteeing the vector's relationships are the same as the nodes and edges' relationships in the knowledge graph.

A widely used graph embedding is TransE [96] embedded graphs by guaranteeing the relationships in the triplet of (head entity, relation, tail entity) in given graphs. The head entity and tail entity indicate the entity nodes connected by a specific relation with a direction. The relation in the triplet is one specific relation connecting two entities. TransE assumes the sum of the head entity's vector, and the relation's vector should be equal to the tail entity's vector to guarantee the relationship of the triplet in the vector space. This assumption can be formulated as:

$$e_t = e_h + e_r \tag{3.1}$$

where e_h and e_t indicate the vectors of head and tail entities, respectively. e_r is the vector of relation. By training the embedding model to satisfy all triplets in the training graph, each entity node and each relation has an identical vector. After TransE, more complicated Trans-series methods were proposed to improve the representations towards the diversity of large-scale knowledge graphs, such as TransH [97] and TransD [98].

In this study, we employed the TransE model pre-trained on the Wikidata knowledge graph to convert the relations in IRPs into embeddings. Specifically, for each relation edge defined in Wikidata, we could use TransE to obtain an identical representation. Thus, we can obtain a sequence of vectors corresponding to the sequence of relations for each IRP.

3.4.2 Model construction

To investigate the effectiveness of IRPs, we proposed a model to utilize IRPs as the IRP slot description in zero-shot slot filling. We followed the one-stage approach to conduct our model, by which the model predicts slot entities from utterances by IOB tags for each given slot. Figure 3.4 shows the model structure.

In the model process, we first converted the IRP patterns into the embedding vector sequences by TransE model for the given slot. Then,

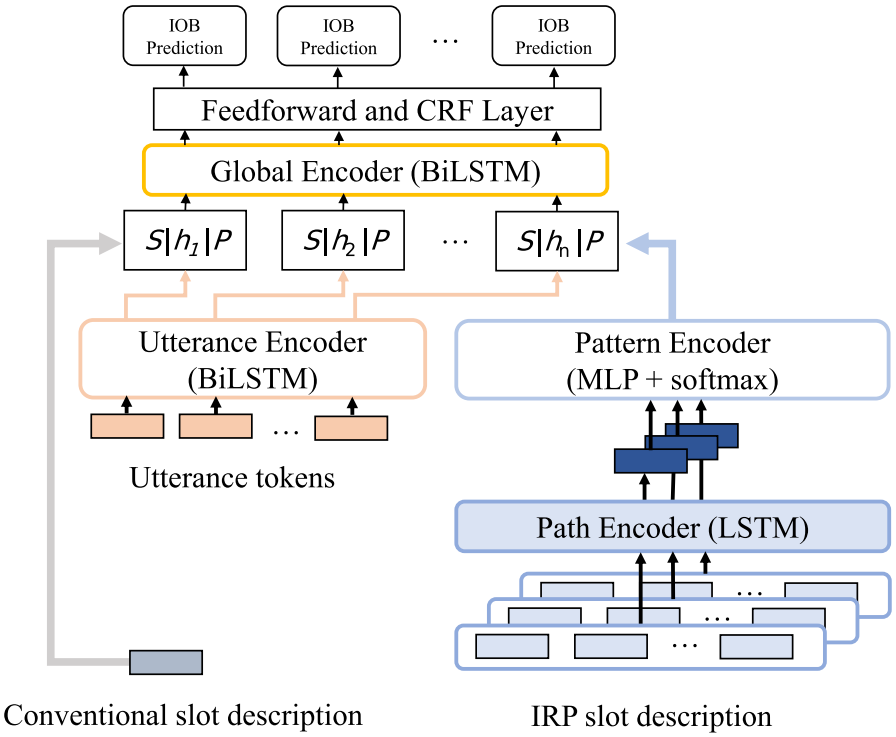

Figure 3.4: Model structure utilizing IRP slot descriptions for zero-shot slot filling

we used a long-short-term memory (LSTM) network as the path encoder to encode the relation path of each IRP pattern separately. The last hidden state of the path encoder was used as path encoding for the corresponding IRP pattern. Next, the pattern encoder takes the path encodings as inputs and used a self-attention mechanism to encode the path encodings into one pattern encoding vector, such a vector represents the combining information of all IRP patterns of the given slot and could be adaptively determine the weights of each IRP pattern through training. The self-attention mechanism is computed as:

$$P = softmax(MLP(V))T * V \tag{3.2}$$

where V is the path encodings of IRP patterns, $MLP(\cdot)$ is a fully connected layer, softmax is the softmax function, T indicates the transpose of $softmax(MLP(V))$, and P is the output pattern encoding vector. Note that we did not use mean pooling of path encodings because we did not know the importance of each path encoding towards various domains and contexts. We used self-attention to let the model learn to decide the importance of each IRP path through the training process.

Besides the IRP slot description that carries implicit semantic information, we also used the conventional slot description as the explicit semantic information. To incorporate the conventional slot description into our model, we first converted the tokens in the conventional slot description into embeddings by word embeddings. Then we performed mean pooling among the tokens to obtain the vector representation of the conventional slot description, denoted as S in the figure. Then, after obtaining the pattern encoding P of the given slot, we concatenate P and S as the combined slot description that considers both explicit and implicit semantic information of a given slot.

Next, the model encodes the given utterance. We converted the tokens of the given utterance into embeddings by word embeddings and used a bidirectional LSTM (BiLSTM) to encode the token embeddings into context encodings. At each time step i, we concatenated the combined slot description with the $i-th$ token's context encoding and input the concatenation into the BiLSTM global encoder to obtain the global encoding. Finally, a conditional random field (CRF) decoder decodes the global encoding into the digits of IOB tags. The model final outputs the sequence of the merged IOB tags for each given slot.

3.4.3 Experiment setting

In this study, we conducted the experiment on the Snips2017 dataset. The Snips2017 dataset contains seven domains with 14484 utterances in total. There are 28 single-domain slots that are only appearing in one domain and 25 multi-domain slots that appear in multiple domains. As introduced in Chapter 2, we followed left-one strategy to evaluate zero-shot slot filling. Specifically, we train a model by setting one domain as the target domain for the zero-shot test and the other six domains as source domains for training. In total, we trained seven models independently by setting each domain in the Snips2017 dataset as the target domain.

In the training process, the data from source domains were merged and divided into the training set and the validation set; while the data from the target domain was used as the test set alone. As our model follows the one-stage paradigm that trains the model to predict slot entities for each given slot, a given slot is generally not mentioned in all training instances of utterances. Accordingly, we followed previous one-stage methods' settings to sample training instances. In particular, for each slot, the training instances were divided into positive instances if the given slot was mentioned in the utterance or negative instances if the given slot was not mentioned. To make a fair comparison with previous studies, we randomly sampled positive and

negative instances in a ratio of 1:3, which is the same as previous studies [20]. We used the slot names as the conventional slot description, which is the universal description and could always be available in different domains since the slot name is necessary for defining the semantic meaning.

For parameter settings, we used the word embeddings based on the neural probabilistic language model [99], which is nnlm-en-dim128, for converting tokens into embeddings. We used the dim-100 TransE [100] model to obtain the relation embeddings for IRPs. We set the LSTM in the path encoder as 128 units, the BiLSTM and LSTM in the utterance encoder, and the global encoder as 200 units, respectively. We employ the cross entropy loss function to compute the loss between the predicted IOB digits and the ground truth IOB labels. The average loss among all tokens in the given utterance is used for optimizing the model parameters. We used the Adam [101] optimizer for optimization with a learning rate of 0.0005. We used the conlleval script [92] to compute the overall slot F1 score to evaluate the model performance on each domain. We computed the slot F1 score for each specific slot to evaluate the model performance on each slot. All experiments were conducted three times to reduce the influence of random initialization. The one-way ANOVA was used to measure the significance of the difference between model performances.

For the baselines, since we follow the one-stage approach in constructing our model, we compare with the baselines from the one-stage methods. Specifically, we use the concept tagger (CT) [19], zero-shot transfer learning (ZAT) [20], and robust zero-shot transfer learning with examples (RZT) [21] as baselines to evaluate the effectiveness of IRPs. As introduced in Chapter 2, ZAT is the best method on the Snips2017 dataset. Therefore, we compare our model with the best method and do not compare with other baselines from the two-stage approach.

We also conducted ablation experiments to clarify the effectiveness of IRP slot descriptions and conventional slot descriptions. To do so, we trained ablation models in the same experiment settings as our original models but using IRP slot descriptions alone and conventional slot descriptions alone. After that, we compare the model performance between the original model and the ablation models.

3.5 Results and discussions

3.5.1 Experimental results and discussions

Table 3.2 shows the results of the model performance on each domain. IRPs is the proposed model using IRP slot descriptions. The Average is the average result over all domains. The underlined numbers indicate the best result of baselines. Bold numbers are the best result for each domain and average. By comparing our model and ZAT, one can see that the proposed model outperformed ZAT in all domains, improving ZAT by 3.61 on average slot F1 score. The ANOVA showed a significant difference between our model and ZAT on the level of $p < 0.001$. These results demonstrated that by taking the implicit semantic information of slots into account, IRP slot descriptions effectively provided transferable information across domains in zero-shot slot filling.

Table 3.2: Results of different model performance for each domain of the Snips2017 dataset

	Baselines			Ours
Domain	CT	ZAT	RZT	IRP
AddToPlaylist	28.94	37.66	26.64	**44.62**
BookRestaurant	24.54	34.05	27.67	**34.62**
GetWeather	42.73	55.82	31.54	**60.05**
PlayMusic	27.18	31.54	24.82	**39.42**
RateBook	20.56	19.47	**25.42**	20.28
SearchCreativeWork	65.95	73.46	65.98	**74.29**
SearchScreeningEvent	24.57	33.05	30.44	**37.04**
Average	33.50	40.72	33.22	**44.33*****

To clarify the effectiveness of IRP slot descriptions on unseen and seen slots, we compared the F1 score on each slot between our model and ZAT and showed the comparison result in Table 3.3. The underlined numbers indicate better results on each slot. The slots with * marks are unseen slots in each domain. To investigate the effectiveness of IRP in more detail, we defined the MSS of unseen slots and the slot type as follows:

MSS of unseen slots: To measure how an unseen slot is semantically similar to the slots in training domains, we defined the max cosine similarity between a given unseen slot and all slots in training domains as the max semantic similarity (MSS). Accordingly, the seen slots do not have MSS. The level of MSS reflects whether the explicit information carried by conventional

slot descriptions could provide transferable information to handle unseen slots or not to some extent. Based on the values of MSS, we divided unseen slots into two categories: the low-MSS unseen slots (MSS < 0.5) and the high-MSS unseen slots (MSS ≥ 0.5).

Slot type: To clarify the effectiveness of IRP on different types of slots, we divided slots into entity-type slots and abstract-type slots based on the slot's semantic meanings. The entity-type slot describes entities or existence in the real world, such as 'album' and 'city'; The abstract-type slot describes concept semantic meanings, such as 'genre' and 'time range.' In Table 3.3, 'E' indicates that the corresponding slots are entity-type slots. 'A' indicates that the corresponding slots are abstract-type slots.

Table 3.3: Results of different methods on each slot

MSS of unseen slots	Type	**Domain/Slots**	Slot F1 score	
		AddToPlaylist	ZAT	Ours
-	E	artist	47.14	55.47
0.73	E	entity name*	8.75	9.26
-	A	music item	84.06	86.75
-	E	playlist	31.75	47.14
0.72	A	playlist owner*	0.43	0.00
		BookRestaurant		
-	E	city	66.64	68.96
-	E	country	69.81	83.92
0.35	A	cuisine*	0.00	0.26
0.45	E	facility*	2.32	0.68
0.65	A	party size description*	2.46	0.00
0.49	A	party size number*	0.06	0.00
0.68	E	poi*	4.09	3.93
0.8	E	restaurant name*	16.30	18.45
0.69	A	restaurant type*	1.92	0.08
0.32	E	served dish*	2.01	7.50
-	A	sort	30.16	14.73
-	A	spatial relation	68.09	60.68
-	E	state	84.13	88.47
-	A	timeRange	45.33	53.95
		GetWeather		
-	E	city	63.38	70.74
0.65	A	condition description*	0.28	0.00
0.42	A	condition temperature*	0.00	4.39
-	E	country	52.52	71.12

0.62	A	current location*	0.00	0.19
0.68	E	geographic poi*	7.61	9.19
-	A	spatial relation	64.83	77.08
-	E	state	79.95	73.48
-	A	time range	83.78	82.76
		PlayMusic		
0.62	E	album*	0.62	8.15
-	E	artist	57.49	67.56
0.52	A	genre*	3.55	2.82
-	A	music item	61.56	69.76
-	E	playlist	6.60	9.37
0.44	E	service*	3.74	15.10
-	A	sort	41.27	46.14
0.35	A	track*	1.14	0.64
0.39	A	year*	0.08	8.37
		RateBook		
0.23	A	best rating*	0.00	0.00
-	E	object name	45.25	35.31
0.69	A	object part of series type*	0.77	1.40
0.65	A	object select*	0.00	18.84
-	A	object type	61.01	43.95
0.45	A	rating unit*	0.42	3.34
0.4	A	rating value*	1.53	10.28
		SearchCreativeWork		
-	E	object name	89.30	87.08
-	A	object type	40.06	51.46
		SearchScreeningEvent		
0.8	E	location name*	37.76	43.80
0.72	E	movie name*	40.41	40.92
0.65	A	movie type*	2.63	15.61
0.9	A	object location type*	21.25	14.75
-	A	object type	0.08	0.00
-	A	spatial relation	65.81	74.35
-	A	time range	84.24	83.76

From Table 3.3, one can see that our model outperforms ZAT on most slots, no matter whether the slots are unseen or seen. To make the comparison clearer and further discussions, we computed the ratio of the slots that our model outperformed ZAT and illustrated the results in sector diagrams.

Figure 3.5 shows the comparison diagrams of low-MSS unseen slots, high-

MSS unseen slots, and seen slots. The blue parts indicate the ratio of the slots that the proposed method outperformed ZAT (R_{ours}), while the orange ones show that ZAT performed better (R_{ZAT}). To evaluate the performance difference, we introduce an improvement ratio R_{imp} as follows:

$$R_{imp} = R_{ours} - R_{ZAT} \tag{3.3}$$

As seen in Figure 3.5, our model achieved an improvement ratio of 40% for low-MSS unseen slots, 28% for seen slots, and 18% for high-MSS unseen slots. These results confirmed that the IRP slot descriptions provided effective transferable information across domains for handling both unseen and seen slots, especially for the unseen slots with less semantic similarity to the slots in source domains, which is hard to deal with by the conventional slot descriptions. On the other hand, our model has a relatively smaller improvement over ZAT for the high-MSS unseen slots. The reason probably is that we utilized the mean pooling of slot tokens' embeddings for conventional slot descriptions in our model, which is not as exquisite as the attention-based representations of slot descriptions used in ZAT.

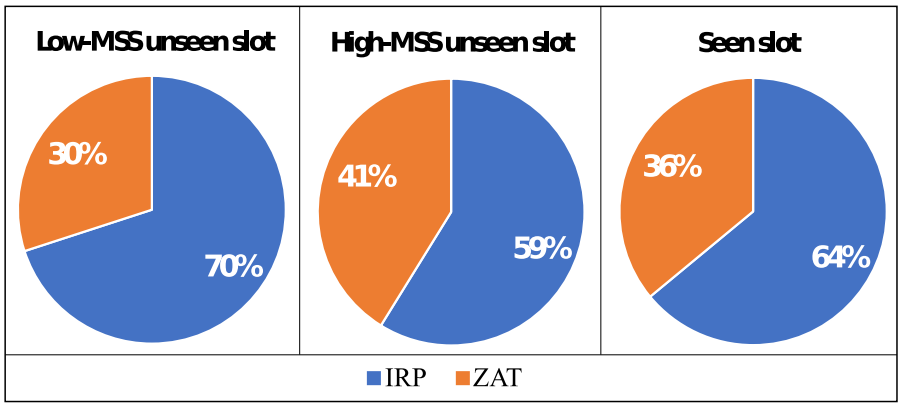

Figure 3.5: Comparison of different methods on different slot categories

Figure 3.6 shows the comparison diagrams of entity-type unseen and seen slots and abstract-type unseen and seen slots. As seen in the figure, our model achieved an improvement ratio of 60% and 50% for unseen and seen entity-type slots, respectively. Moreover, our model achieved an improvement ratio of 6% and 20% for unseen and seen abstract-type slots, respectively. These results demonstrated that the IRP slot descriptions provided transferable information effectively in handling entity-type slots. On the other hand, our model achieved relatively smaller improvements on abstract-type slots. The reason is that unlike entity relations obtained from the commonsense

knowledge graph Wikidata, abstract-type slots are not so effective in having the inference path.

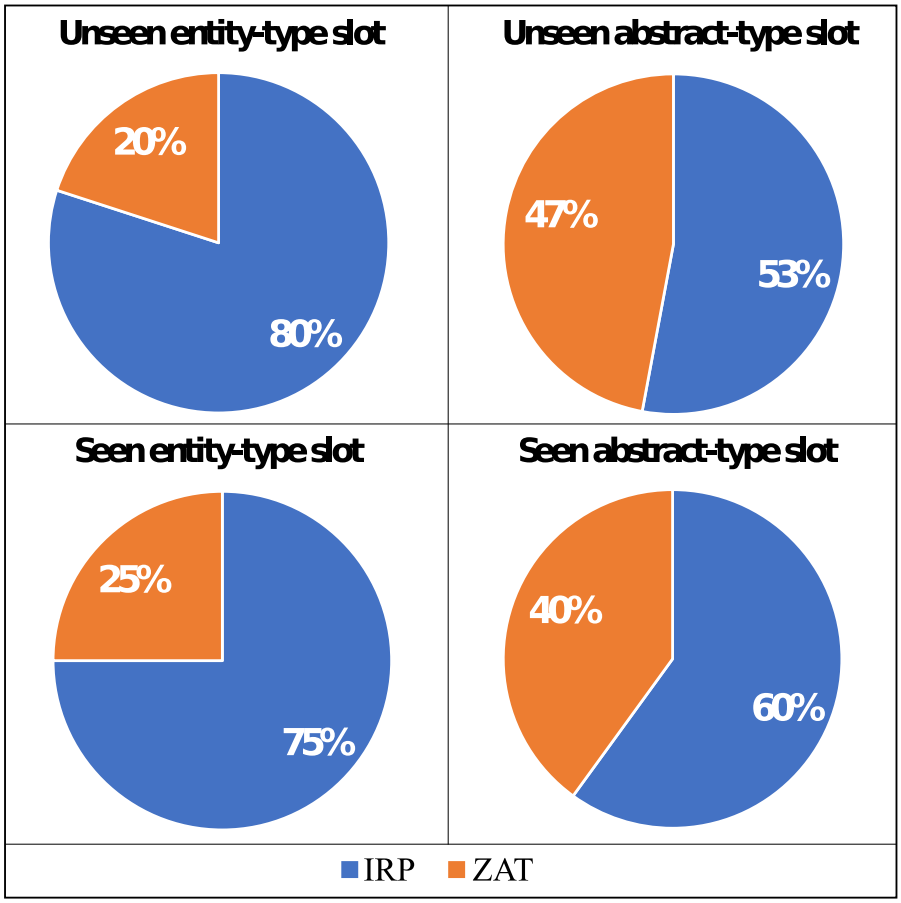

Figure 3.6: Comparison of different methods on different types of slots

3.5.2 Ablation studies

Next, Table 3.4 shows the comparison results of the ablation experiments. In the table, Origin indicates the performance of the original model. IRP alone and conventional alone are the performance of the ablation models using IRP slot descriptions alone and using conventional descriptions alone, respectively. As seen in the table, without the explicit slot information carried by the conventional slot descriptions, using IRP slot descriptions alone achieved 23.88 on slot F1 score, which is over half of the slot F1 score of the using conventional slot descriptions alone. This result demonstrated that without using the explicit semantic information that is the basic cue to

correlate domains, using the implicit semantic information carried by IRPs alone could still provide effective transferable information across domains for zero-shot slot filling. The results also show that adding IRP slot descriptions to the conventional slot descriptions significantly improved the model performance on all domains, improving the average F1 score by 4.16. These results demonstrated that the IRP slot descriptions and the conventional slot descriptions have a complementary effect on zero-shot slot filling.

Table 3.4: Results of ablation model performance on each domain of the Snips2017 dataset

Domain	Origin	IRP alone	Conventional alone
AddToPlaylist	44.62	43.72	37.93
BookRestaurant	34.62	12.42	33.56
GetWeather	60.05	35.72	59.32
PlayMusic	39.42	22.72	30.16
RateBook	20.28	6.92	18.80
SearchCreativeWork	74.29	33.11	68.08
SearchScreeningEvent	37.04	12.57	33.36
AVER	44.33	23.88	40.17

3.6 Summary

In this chapter, we proposed the IRP for alleviating the domain shift problem of semantically dissimilar unseen slots. As described above, using IRP slot descriptions improved the zero-shot slot filling on handling both unseen and seen slots. Although we used IRPs with the one-stage approach, IRPs can also be used with two-stage approach since IRPs describe the property of slots and can provide transferable information for slot type prediction in stage two of the two-stage approach.

However, the absolute improvements were not high on unseen slots. Furthermore, obtaining IRPs are not flexible in practice. Specifically, in the process of extracting IRPs on the Snips2017 dataset via Wikidata described in Section 3.2, we have already encountered the situations that some slots or values could not be identified as entities. This is partially because the characteristics of Wikidata, since Wikidata is a commonsense knowledge graph that mainly describes entities and the relations between entities in the real world and is not effective to identify the slots and values that have concept semantic meaning. Another reason is that some slots and values are the combination of entities, such as 'object select' and 'location name',

those slots and values cannot be identified as an entity or a concept by the knowledge graph, so that it is hard to extract exact IRPs for such slots. In addition, due to the interpretability limitation of the one-stage approach described in Chapter 2, it is hard to analyze and explore intrinsic representations in zero-shot slot filling.

Chapter 4

Multi-relation-based representation

4.1 Objective

To overcome the limitations of IRPs and the one-stage approach to deal with domain shift problems in zero-shot slot filling, we moved our sight to the two-stage approach of zero-shot slot filling. Since the correct prediction for slot entities is the premise for improving zero-shot slot filling, we focused on mining the intrinsic representation to alleviate the domain shift problem of different context distributions in slot entity predictions. Specifically, we aimed to improve slot entity predictions from the context corresponding to unseen slots and the context corresponding to seen slots. Meanwhile, the intrinsic representation should be flexible that can be obtained easily. For this purpose, this chapter introduces the multi-relation-based representation for improving slot entity predictions to improve zero-shot slot filling.

4.2 Previous representations for slot entity predictions

Zero-shot slot filling approaches predicted slot entities and slot types separately to avoid retraining the model for handling unseen slots. One-stage methods [19, 20, 21] predicted slot entities for each slot type, while two-stage methods [22, 23, 83] predicted slot types based on the predictions of slot entities. Accordingly, the correctness of slot entity predictions is the premise to guarantee the performance of zero-shot slot filling.

As introduced in Chapter 2, in one-stage methods, concept tagger (CT) realized the slot entity prediction by combining the contextual information and the slot description's information at each time step. Zero-shot adaptive transfer learning (ZAT) used an attention mechanism to obtain a context-aware slot description and realized the slot entity prediction by considering

contextual information and context-aware slot description's information. Robust zero-shot slot filling tagger with examples (RZT) combined the information of example slot values at each time step as additional semantic information to consider contextual information, slot description's information, and examples' information.

Accordingly, one-stage methods combined contextual information with slot type's information. The slot entity prediction process for one-stage methods could be briefly formulated as follows:

$$IOB^k = Dec(Enc(e_1^c, e_{slot_k}), Enc(e_2^c, e_{slot_k}), ..., Enc(e_i^c, e_{slot_k}), ...), slot_k \in N_s \tag{4.1}$$

here, N_s is the slot set of a domain. e_{slot_k} is the representation of the $k-th$ slot type, which can be obtained from the mean or sum pooling of slot description token's embeddings. e_i^c is a context vector of the $i-th$ token in a given utterance. IOB^k is the sequence of the IOB tags corresponding to the $k-th$ slot type. $Enc(\cdot)$ is an encoder to encode the combined information of e_i^c and e_{slot_k}, and $Dec(\cdot)$ is a decoder to obtain the IOB tag distribution. Previous studies used RNN-based structure, such as LSTM, as the encoder and used feedforward layers or CRF layer as the decoder.

In one-stage methods, the $E(e_i^c, e_{slot_k})$ is a slot-type-based representation that reflects the specific characteristics of slot entities regarding to the $k-th$ slot type. It can provide transferable information to predict slot entities corresponding to specific slot types in target domains. However, one-stage methods were argued to have less capability to capture the complete slot entity for a given slot type in new domains [22] because they learn the slot entities for corresponding slot types.

Two-stage methods [22] were proposed to learn context-based representations to improve the prediction of complete slot entities for zero-shot slot filling. A context-based representation captures the general characteristics of slot entities according to various context environments without considering specific slots. Specifically, the two-stage methods so far [22, 23, 83] all used the same algorithm to realize the slot entity prediction by considering contextual information alone. The prediction process of these two-stage methods can be formulated as follows:

$$IOB = Dec(e_1^c, e_2^c, ..., e_i^c, ...) \tag{4.2}$$

here, IOB is the sequence of the IOB tags, e_i^c is the context vector of the $i-th$ token in a given utterance. $Dec(\cdot)$ is a decoder. Previous studies usually used CRF for the decoder to consider the contextual dependency of context vectors.

In the two-stage methods, the context vector e_i^c is a context-based representation that reflects the general characteristics of slot entities without considering specific slots. However, because of domain shift problems, for the context corresponding to the unseen slots and the context corresponding to the differently explained seen slots, it is hard to predict correct slot entities based on the general slot entity patterns learned in training without using any additional information.

For the problems mentioned above, we aimed to mine a representation that captures general slot entities in diverse context environments and still be able to deal with the unseen contexts in new domains even if the contexts correspond to unseen slots or differently explained seen slots. Meanwhile, considering the limitation of IRPs mentioned above, the representation should be obtained easily based on slots and utterances to guarantee flexibility.

4.3 Multi-relation-based representation

For our purpose, we revisited the representations for slot entity predictions in previous studies. In an one-stage method, the slot-type-based representation carried entity-type relations between the slot entity and a specific slot type. While the context-based representation in the two-stage method reflected the entity-context relation between a given slot entity and a variety of contextual environments among different domains. To tackle the domain shift problem mentioned above, it would be better to combine the advantages of the one-stage and two-stage methods. The context-based representation was effective on capturing complete slot entities. The slot-type-based representation could provide the similarities between domains from the aspect of slots, thus can be expected to provide additional information for capturing slot entities from unseen contexts in new domains. Accordingly, we proposed a multi-relation-based representation to take both the entity-type and entity-context relations into account and use this representation for the slot entity prediction. The multi-relation-based representation is obtained by averaging the representations of entity-type relations for all slot types in a given domain, which is formulated as follows:

$$repr_i = \frac{1}{|N_s|} \sum_{k=1}^{|N_s|} Enc(e_i^c, e_{slot_k}), slot_k \in N_s \tag{4.3}$$

$$IOB = Dec(repr_1, repr_2, ..., repr_i, ...) \tag{4.4}$$

where $|N_s|$ is the size of the slot set N_s.

In eq 4.3, $repr_i = \frac{1}{|N_s|}\sum_{k=1}^{|N_s|} Enc(e_i^c, e_{slot_k})$ is the multi-relation-based representation. This representation takes the slot type information into account and captures the general slot entity pattern by involving all possible slot types in a given domain but not a specific one. The multi-relation-based representation provides a unified representation of the general and specific characteristics of slot entities. The general characteristic guarantees the capacity on capturing complete slot entities, while the specific characteristic provide additional information for capturing the slot entities from unseen contexts. Thus, this representation potentially offers transferable information across domains. Meanwhile, the proposed representation can be obtained from utterances and slots. Thus it guarantees the flexibility of obtaining and overcoming the limitations of IRPs. Consequently, the proposed representation can be expected to improve zero-shot slot filling by alleviating domain shift problems in the slot entity prediction.

4.4 Experiments

4.4.1 Model construction

Since the multi-relation-based representation is used for predicting the slot entity without considering specific slots, it is suitable to construct a two-stage model to use the proposed representation for one stage of slot entity predictions. Accordingly, we constructed a two-stage model, in which stage one used the proposed representation for predicting slot entities, and stage two used the same method as Coach [22] to predict slot types based on the predicted slot entities.

Figure 4.1 shows the model structure. In stage one, we first converted the tokens in the given utterances into embedding representations. We also converted the tokens of the slot descriptions into embedding representations and obtained a slot type vector representation e_{slot_i} for each slot type's description via pooling. In this chapter, we investigated the effectiveness of different pooling methods, which were mean pooling and sum pooling. The details will be described in the next section. Then, we used a BiLSTM layer as an encoder to encode the input utterance into context vectors. At each time step, the context vector and the $k-th$ slot type's vector representation are concatenated as the input to the second BiLSTM layer. The second BiLSTM layer calculates the fusion encoding of the contextual information and the $k-th$ slot type's information. Following the definition of the proposed representation, fusion encodings are obtained for all slot types in a given domain. Then we computed the multi-relation-based representation for the

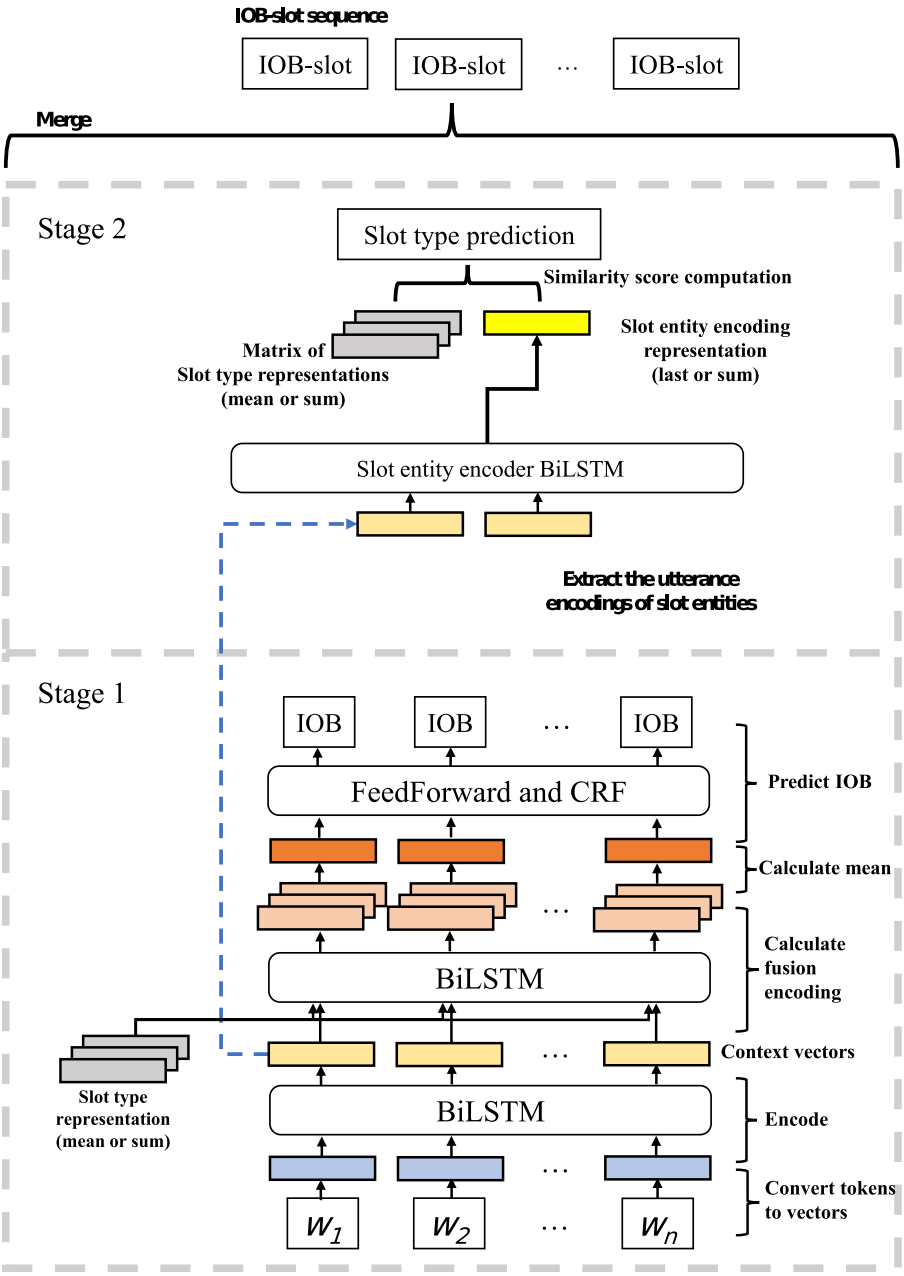

Figure 4.1: Two-stage model structure using the multi-relation-based representation for slot entity predictions in stage one

$i-th$ token as the average of the fusion encoding as follows:

$$repr_i = \frac{1}{|N_s|} \sum_{k=1}^{|N_s|} Enc(e_i^c, e_{slot_k}) \tag{4.5}$$

finally, we inputted the sequence of all token's multi-relation-based representation into the CRF layer to obtain the IOB tags for the slot entity prediction.

In stage two, we followed the similarity-based method to predict slot types for the slot entities from stage one. Specifically, we first extracted the slot entities based on the IOB tags. To do so, we extract the token with the B tag as a beginning of a slot entity and the tokens with I tags following the beginning token as the inner of the slot entity. Then we extracted the context vectors of the slot entity from the first BiLSTM encoder, and we used a BiLSTM to encode these context vectors into entity encodings. In this chapter, we also investigated the effectiveness of different ways to obtain entity encodings, which will be described in the next section. Finally, we computed the softmax similarity score between the slot entity and all slot types in the given domain as follows:

$$sim_k = e_{se} \cdot e_{slot_k}, slot_k \in N_s \tag{4.6}$$

$$slot_{pred} = argmax(softmax(sim_1, sim_2, ..., sim_k, ...)) \tag{4.7}$$

where e_{se} is the entity encoding of the slot entity, e_{slot_k} is the vector representation of the $k-th$ slot type, sim_k is the similarity score between e_{se} and e_{slot_k}. $softmax$ is the softmax function that converts the similarity scores into a probability distribution. $argmax$ is the function to choose the slot type with the highest softmax similarity score. $slot_{pred}$ is the slot type prediction.

After the two stages, the model merges the slot entity predictions and slot type predictions into a final output formed as an IOB-slot tag sequence.

4.4.2 Comparison of slot type vector representations and slot entity encoding representations

To achieve better performance in zero-shot slot filling, previous studies proposed several different forms for both the slot type vector representation e_{slot_i} and for the slot entity encoding representation e_{se}. However, there is less knowledge about the effect of those representations. In this chapter, to investigate the characteristics and efficiency of each representation, we compared the performance of various combinations of the slot type vector representations and slot entity encoding representations.

For the slot type vector representation, some studies [19, 21] showed that the mean embedding of slot description tokens obtained by mean pooling is effective in zero-shot slot filling. While other studies proposed a sum embedding of the description tokens obtained by sum pooling, as in the case of [22]. For brevity, we hereafter refer to the former kind of representation

as 'mean-slot' and the latter as 'sum-slot.' If a slot description contains m tokens, we can formulate the slot type vector representations as follows:

$$repr^{slot}_{mean} = \frac{1}{m}\sum_{i=1}^{m} emb_i, i \in [1, m], \tag{4.8}$$

$$repr^{slot}_{sum} = \sum_{i=1}^{m} emb_i, i \in [1, m]. \tag{4.9}$$

where $repr^{slot}_{mean}$ and $repr^{slot}_{sum}$ are the mean-slot and sum-slot representations, respectively. emb_i is the embedding of the $i-th$ token.

The slot entity encoding representation is the encoding representation of a given slot entity text sequence. The last time step of the sequence encoding vectors [102] and the sum of each time step vector [22] were widely used for encoding representations. For brevity, we hereafter refer to the former kind of representation as 'last-enc' and the latter as 'sum-enc.' If a slot entity contains n tokens, the representations can be formulated as follows:

$$h_i = E(h_{i-1}), \tag{4.10}$$

$$repr^{enc}_{last} = h_n, \tag{4.11}$$

$$repr^{enc}_{sum} = h_1 + ... + h_n. \tag{4.12}$$

here, $repr^{enc}_{last}$ and $repr^{enc}_{sum}$ are the last-enc and sum-enc representations, respectively. h_i is the encoding vector of the $i-th$ token in the given slot entity, and $E(\cdot)$ is an encoder.

4.4.3 Experiment setting

In this chapter, we conducted our experiments on three multi-domain datasets to evaluate the effectiveness of the proposed multi-relation-based representation, Snips, SGD, and MultiWOZ 2.3 datasets. In addition, we also conducted the experiment on the Snips2017 dataset to compare the performance between the multi-relation-based representation to IRPs that introduced in the last chapter.

Since zero-shot slot filling aims to process the input utterances of users, we performed data cleanings to obtain the utterances from the user speaker alone. To do so, we removed the utterances from the system speaker in the SGD and MultiWOZ 2.3 datasets. Then, we removed the utterances without any slot, as the slots are the key objects for zero-shot slot filling. The Snips dataset contains utterances only from the user side, and all utterances mention at least one slot. Accordingly, the Snips dataset was fully used

Table 4.1: Statistics of datasets after data cleaning

Dataset	Domains	Utterances	SD slots	MD slots
Snips	7	14484	28	25
SGD	20	70166	63	38
MultiWOZ 2.3	6	43591	6	24

in this research. After the data cleaning, we found that two domains in the MultiWOZ 2.3 dataset (the 'Hospital' and 'Police' domains) contained much less data than others. For meaningful evaluations, we merged these two domains into a single domain called 'Others.' After data cleaning, Snips contains seven domains, MultiWOZ 2.3 contains six domains, and SGD contains 20 domains.

To show the details of each dataset clearly, Table 4.1 lists domains, utterances, SD slots, and MD slots in each domain for Snips, SGD, and MultiWOZ 2.3 datasets, respectively.

Similar to the experiments in Chapter 3, this chapter also follows the same training process as previous studies. Specifically, we conducted our experiments on each dataset separately. For each dataset, we set one domain as the zero-shot target domain and other domains as training domains. For each zero-shot target domain, we trained a model on the data of training domains and evaluated the model performance on the zero-shot domain. Accordingly, we trained seven models for the Snips dataset, six models for the MultiWOZ 2.3 dataset, and 20 models for the SGD dataset.

In the training process, we merged all data from the training domains as the training set for training. Furthermore, we split the data from the zero-shot targe domain into a validation set to choose the best model and a test set to evaluate the model performance, which was the same approach as in the previous works [22]. In particular, there are two splitting situations based on the instances in the zero-shot target domain: (1) Most domains have more than 600 instances. For those domains, we followed the setting in previous works [22] to use the first 500 instances in the zero-shot target domain as the validation set and use the remaining data as the test set; (2) Some domains have less than 600 instances, such as the 'Message' domain in the SGD dataset. To perform a meaningful evaluation on those domains, we used the first 40% instances as the validation set and the remaining 60% as the test set.

For the parameter settings, the two BiLSTMs in stage one were both set as one layer with 256 hidden units. The encoder BiLSTM in stage two was set as two layers with 256 hidden units in each layer. We used a 400-

dimension word representation to obtain the embeddings for tokens. The 400-dimension representation is the concatenation of a 300-dimension word embedding and a 100-dimension character embedding. The word embedding was obtained from Fast-Text [103, 104], and the character embedding was obtained from torchtext [105]. We used the CRF loss to compute the loss for the IOB prediction in stage one. We used the cross-entropy loss to compute the loss among slot type predictions in stage two. The Adam optimizer was used for optimization with a learning rate of 0.0005. All experiments were conducted three times to reduce the influence of random initialization. One-way ANOVA was used to measure the significance level of difference between methods.

For evaluation metrics, to precisely evaluate the effectiveness of the proposed multi-relation-based representation on the slot entity prediction, we use the Conlleval script [92] to compute the IOB F1 score that only computes the accuracy of slot entities without considering specific slots. Then, to investigate the model performance on predicting the slot entities corresponding to seen and unseen slots, we proposed the prediction accuracies of the seen slots' entities and unseen slots' entities. Note that we computed accuracies rather than the IOB F1 score for separate evaluations on seen and unseen slots. This is because we can pick the ground-truth slot entities corresponding to seen and unseen slots from utterances and compute the recall, but we cannot compute the precision since the predictions are not specified to seen or unseen slots.

After that, to investigate how the proposed representations contribute to zero-shot slot filling by contributing to slot entity predictions, we computed the slot F1 score to evaluate the performance of zero-shot slot filling on each domain. We also computed the slot F1 score for unseen and seen slots to evaluate the model performance on each slot.

We used the two-stage method Coach [22] as the baseline to precisely evaluate the effectiveness of the proposed representation used in stage one, because stage two of our model is the same as Coach. We also used the best methods on each dataset as baselines. As introduced in Chapter 2, the best methods on Snips, MultiWOZ 2.3, and SGD datasets are RZT, CT, and Coach, respectively. Accordingly, we compare two baselines on Snips and MultiWOZ 2.3 datasets, and we only compare with Coach on the SGD dataset.

4.5 Results and discussions

4.5.1 Experimental results and discussions

Tables 4.2, 4.3, and 4.4 show the IOB F1 score results of different methods for the Snips, MultiWOZ 2.3, and SGD datasets, respectively. Due to the limitations of space, we cannot show the methods' results for each domain in a single row for Snips and MultiWOZ 2.3 datasets. Therefore, we split the table into two parts for these two datasets, in which (a) lists the results from baseline methods, and (b) lists the results from the proposed method. In the table, domain indicates the model performance on each zero-shot target domain. Underlined numbers are the best results of baselines on each domain and average on Snips and MultiWOZ 2.3 datasets. Bold numbers indicate the best results on each domain and average, respectively. The performance of our model were obtained with different slot type vector representations and slot entity encoding representations. For brevity, we denote these four combinations as follows: 'mean-last' is the combination of the mean-slot slot

Table 4.2: IOB F1 scores on the Snips dataset

(a) Results of the baseline methods				
Domain	Coach		RZT	
AddToPlaylist	61.26		54.45	
BookRestaurant	58.43		53.11	
GetWeather	61.94		64.52	
PlayMusic	50.55		45.43	
RateBook	31.66		32.24	
SearchCreativeWork	51.27		38.51	
SearchScreeningEvent	39.92		32.59	
Average	50.72		45.84	
(b) Results of the proposed methods				
Domain	mean-last	mean-sum	sum-last	sum-sum
AddToPlaylist	**63.17**	61.75	62.13	60.93
BookRestaurant	**64.39**	62.55	64.31	62.82
GetWeather	**70.14**	67.52	66.07	66.66
PlayMusic	**68.16**	64.58	66.72	66.13
RateBook	29.78	30.96	**33.81**	29.20
SearchCreativeWork	50.93	52.72	51.74	**54.39**
SearchScreeningEvent	**50.60**	50.18	47.88	48.74
Average	**56.74****	55.75*	56.10**	55.55**

Table 4.3: IOB F1 scores on the MultiWOZ 2.3 dataset

(a) Results of the baseline methods

Domain	Coach	CT
Attraction	68.25	66.10
Hotel	69.65	71.66
Restaurant	76.05	72.47
Taxi	54.60	58.51
Train	73.62	**82.27**
Others	58.59	**59.66**
Average	66.79	68.44

(b) Results of the proposed methods

Domain	mean-last	mean-sum	sum-last	sum-sum
Attraction	65.49	64.67	72.49	**72.65**
Hotel	70.36	**72.89**	69.72	72.26
Restaurant	74.08	74.61	**76.05**	74.80
Taxi	65.31	**66.61**	63.74	66.25
Train	76.85	78.24	77.12	76.88
Others	56.47	58.20	58.75	59.19
Average	68.09	69.20	69.64	**70.34**

type vector representation and last-enc slot entity encoding representation. Similarly, 'mean-sum' is the combination of the mean-slot slot type vector representation and the sum-enc slot entity encoding representation. On the other hand, 'sum-last' and 'sum-sum' are the sum-slot representation with the last-enc and sum-enc representations, respectively. The '*' and '**' beside the average results are the significance level based on one-way ANOVA between the better baseline and the corresponding model, which indicate the level of $p < 0.05$ and $p < 0.01$, respectively.

As seen in the tables, Coach achieved an average IOB F1 score of 51.53, 66.79, and 89.52 on Snips, MultiWOZ 2.3, and SGD datasets, respectively. From Table 4.2, one can see that our model using mean-last representations achieved the best performance on the Snips dataset and improved the baseline by 6.02 of the IOB F1 score. From Tables 4.3 and 4.4, one can see that our model using sum-sum representations achieved the best performance on MultiWOZ 2.3 and SGD datasets, improved the baseline by 3.55 and 0.64 of average IOB F1 score on MultiWOZ 2.3 and SGD datasets, respectively. These results demonstrated that by capturing the general and specific characteristics of slot entities, the proposed multi-relation-based representation alleviated the domain shift problem on the slot entity prediction.

Besides comparing with Coach, the comparison between out best model

Table 4.4: IOB F1 scores on the SGD dataset

	Baseline	Ours			
Domains	Coach	mean-last	mean-sum	sum-last	sum-sum
Alarm	**85.72**	84.47	82.97	81.80	83.12
Banks	88.81	91.82	**94.02**	89.85	90.20
Buses	99.36	99.19	99.34	**99.39**	99.24
Calendar	82.89	83.03	82.79	**83.16**	83.04
Events	90.06	91.23	91.91	90.89	**91.94**
Flights	97.43	97.34	97.63	**98.15**	98.04
Homes	96.58	97.26	97.13	97.26	**97.46**
Hotels	81.75	81.09	81.36	81.10	**82.82**
Media	90.72	92.10	92.31	**93.47**	93.05
Messaging	**88.10**	85.02	87.32	83.41	85.50
Movies	89.87	89.54	89.56	**90.15**	90.05
Music	53.79	52.66	56.38	52.27	**59.22**
Payment	94.75	95.51	**96.48**	94.84	95.51
RentalCars	98.48	98.55	98.90	98.95	**98.99**
Restaurants	77.38	**81.65**	80.74	80.53	81.26
RideSharing	**81.45**	76.62	78.48	76.31	79.88
Services	96.64	**98.24**	97.66	98.21	97.39
Trains	**99.58**	99.51	**99.58**	99.44	**99.58**
Travel	98.45	97.05	98.37	**98.59**	98.26
Weather	98.66	**98.88**	98.87	98.39	98.59
Average	89.52	89.54	90.09	89.31	**90.16**

mean-last and RZT on the Snips dataset showed that mean-last outperformed RZT by 10.9 on the IOB F1 score. By comparing our best model sum-sum with the best method CT on the MultiWOZ 2.3 dataset, one can see that sum-sum outperformed CT by 1.9 on the IOB F1 score. These results confirmed that the proposed representation is effective on alleviating domain shift problems.

Note that our model achieved a relatively small improvement on the SGD dataset. This is because the seen slots in the SGD dataset are usually not explained differently due to the unseen context in different domains, unlike that in the other two datasets. For instance, in the Snips dataset, the slots 'object name' and 'object type' are explained differently in the Rate Book domain and Search Creative Work domain. In the Rate Book domain, the slot 'object name' indicates a book the user wants to rate for, and the slot 'object type' indicates the type of a book. On the other hand, in the Search Creative Work domain, the slot 'object name' indicates the name of a movie

Table 4.5: Prediction accuracies on different datasets. (a) shows the prediction accuracies on the slot entities corresponding to unseen slots; (b) shows the prediction accuracies on the slot entities corresponding to seen slots

(a) Prediction accuracies on the slot entities corresponding to unseen slots

Dataset	Snips		MultiWOZ 2.3		SGD	
Baselines	Coach	40.31%	Coach	28.82%	Coach	85.52%
	RZT	14.91%	CT	23.08%		
Ours	mean-last	**49.59%****	mean-last	21.92%	mean-last	86.04%
	mean-sum	47.45%*	mean-sum	**29.05%**	mean-sum	**86.74%**
	sum-last	47.17%**	sum-last	28.92%	sum-last	84.58%
	sum-sum	46.70%**	sum-sum	26.83%	sum-sum	85.88%

(b) Prediction accuracies on the slot entities corresponding to seen slots

Dataset	Snips		MultiWOZ 2.3		SGD	
Baselines	Coach	58.75%	Coach	71.85%	Coach	89.36%
	RZT	52.85%	CT	72.53%		
Ours	mean-last	**65.57%*****	mean-last	73.17%	mean-last	89.97%
	mean-sum	64.91%***	mean-sum	73.85%	mean-sum	90.25%
	sum-last	64.48%**	sum-last	73.78%	sum-last	89.52%
	sum-sum	65.14%**	sum-sum	**74.62%***	sum-sum	**90.55%***

or a song, and the slot 'object type' indicates the type of creative work. While in the SGD dataset, most seen slots do not change the semantic meanings in different domains. Therefore, the proposed representation's advantage in handling the slot entity corresponding to the differently explained seen slots was not fully brought out on the SGD dataset.

Then, we investigated model performance on the slot entities corresponding to unseen and seen slots. Table 4.5 shows the prediction accuracies of different methods on all three datasets. '*', '**', and '***' indicate the corresponding result has significant difference to the better baseline at the level of $p < 0.05$, $p < 0.01$, and $p < 0.001$. As seen in Table 4.5 (a), Coach outperformed than RZT on the prediction accuracies of unseen and seen for Snips and MultiWOZ 2.3 datasets. Coach also outperformed CT on the prediction accuracy of seen slots for the Snips dataset. While CT outperformed Coach a bit on the prediction accuracy of seen slot for the MultiWOZ 2.3 dataset. These results showed that the two-stage method Coahc is better than one-stage methods on most situations of slot entity predictions, which is consistent with the arguement in the paper of Coach [22].

By comparing our models and baselines, one can see that mean-last

Table 4.6: Zero-shot slot filling performance on the Snips dataset

	Baselines		Ours
Domain	Coach	RZT	sum-sum
AddToPlaylist	47.30	50.27	**51.56**
BookRestaurant	31.30	34.52	**36.90**
GetWeather	45.97	**61.44**	52.66
PlayMusic	29.99	**39.70**	38.66
RateBook	9.60	25.01	**14.78**
SearchCreativeWork	50.55	38.40	**53.90**
SearchScreeningEvent	19.19	20.15	**25.63**
Average	33.42	38.50	**39.16**

combination is the best model for predicting unseen slots' entities for the Snips dataset, improving the better baseline Coach by 9.28% on prediction accuracy. The mean-sum combination is the best model for predicting unseen slots' entities for MultiWOZ 2.3 and SGD datasets, improving the better baseline Coach by 2.30% and 1.22% on MultiWOZ 2.3 and SGD datasets, respectively. As seen in Table 4.5 (b), the mean-last combination is the best model for predicting seen slots' entities for the Snips dataset, improving Coach by 6.82% on prediction accuracy. The sum-sum combination is the best model for predicting seen slots' entities for MultiWOZ 2.3 and SGD datasets, improving the better baselines CT of the MultiWOZ 2.3 dataset and Coach of the SGD dataset by 2.09% and 1.19%, respectively. These results demonstrated that capturing general and specific characteristics effectively alleviated the domain shift problem of predicting the slot entities corresponding to unseen and seen slots.

To identify the most effective combination of slot type vector representations and slot entity encoding representations, we performed a one-way ANOVA among four combinations on each dataset. As a consequence, the one-way ANOVA showed no significant difference among four combinations ($p > 0.05$) in each dataset. Therefore, we then compared the average performance over three datasets for the four combinations to choose the most effective one. According to the results in Tables 4.2, 4.3, and 4.4, the average performance of mean-last, mean-sum, sum-last, and sum-sum are 71.46, 71.68, 71.68, and 72.02. Therefore, the sum-slot + sum-enc combination outperforms others on the average performance over three datasets, so it is the most effective combination for the model structure of this chapter. Following the results, we chose the model of sum-sum combination as our representative model and used it for subsequential analysis.

Table 4.7: Zero-shot slot filling performance on the MultiWOZ 2.3 dataset

	Baselines		Ours
Domain	Coach	CT	sum-sum
Attraction	66.07	65.94	**69.97**
Hotel	64.24	**66.89**	64.99
Restaurant	70.40	71.06	**72.40**
Taxi	39.17	**52.86**	40.97
Train	62.66	**80.19**	68.88
Others	54.75	**58.49**	56.74
Average	59.55	**65.91**	62.32

Next, we evaluated how the improvement by the proposed representation contributes to zero-shot slot filling. Tables 4.6, 4,7, and 4.8 show the model performance of zero-shot slot filling on Snips, MultiWOZ 2.3, and SGD datasets, respectively. As seen in the tables, the baseline method Coach achieved an average slot F1 score of 33.42, 59.55, and 68.72 of average slot F1 score on Snips, MultiWOZ 2.3, and SGD, respectively. Our model using sum-sum representation outperformed Coach by 5.74, 2.77, and 0.87 on Snips, MultiWOZ 2.3, and SGD datasets, respectively. As our model using the same stage two for slot type predictions as used in Coach, these results demonstrated that by alleviating the domain shift problem on the slot entity prediction, the proposed representation improved zero-shot slot filling performance.

To investigate the consistency between the improvements in the slot entity prediction and zero-shot slot filling, we computed the prediction error reduction rate R_e to show the relative improvement compared to the baseline method as follows:

$$R_e = (F1_{ours} – F1_{baseline})/F1_{baseline} \tag{4.13}$$

where the $F1_{ours}$ and $F1_{baseline}$ are the average performance of our representative model and the baseline method. The difference $F1_{ours} – F1_{baseline}$ is the improvement of our model. From Tables 4.2, 4.3, and 4.4, we can obtain the R_e of the sum-sum model on the slot entity prediction, which are 9.52%, 5.32%, and 0.07% on Snips, MultiWOZ 2.3, and SGD datasets, respectively. From Tables 4.6, 4.7, and 4.8, we can obtain the R_e of the sum-sum model on zero-shot slot filling, which are 17.18%, 4.65%, and 1.27% on the Snips, MultiWOZ 2.3, and SGD datasets, respectively. Accordingly, these results showed that the tendency of error reduction in the slot entity prediction was similar to the tendency of improvement in zero-shot slot filling. Therefore, the results demonstrated that due to the improvements in the slot entity

Table 4.8: Zero-shot slot filling performance on the SGD dataset

	Baseline	Ours
Domain	Coach	sum-sum
Alarm	**85.72**	83.12
Banks	**58.37**	31.41
Buses	**34.47**	34.30
Calendar	81.51	**82.17**
Events	64.24	**69.06**
Flights	44.21	**60.61**
Homes	95.48	**96.49**
Hotels	**44.86**	38.27
Media	51.13	**54.83**
Messaging	**73.50**	71.25
Movies	64.93	**70.17**
Music	31.52	**39.92**
Payment	82.33	**89.85**
RentalCars	37.79	**38.20**
Restaurants	59.21	**65.03**
RideSharing	**81.45**	79.88
Services	95.36	**95.92**
Trains	91.11	**94.44**
Travel	**98.45**	98.26
Weather	**98.66**	98.59
Average	68.72	**69.59**

prediction, our model improved the baseline method consistently on zero-shot slot filling.

By comparing with the best methods of RZT, CT, and Coach on Snips, MultiWOZ 2.3, and SGD datasets, one can see that our model outperformed the best method of RZT and Coach on almost domains and achieved better performance on average. On the other hand, our model did not outperform CT, which is the best method on the MultiWOZ 2.3 dataset. We consider the limitation was on stage two of slot type predictions since our model was shown better than CT on slot entity predictions from Table 4.3 and Table 4.5, stage two did not bring out the improvements of stage one for zero-shot slot filling. Accordingly, improving slot type prediction is also necessary to improve the overall zero-shot slot filling.

Then, we analyzed the model performances on unseen and seen slots to investigate how the improvements in the slot entity prediction contribute to

Table 4.9: Average slot F1 score for unseen and seen slots on each dataset. (a) shows slot F1 scores on unseen slots; (b) shows slot F1 scores on seen slots

(a) Slot F1 score on unseen slots

Dataset	Snips		MultiWOZ 2.3		SGD	
Baselines	Coach	**5.23**	Coach	16.82	Coach	41.00
	RZT	5.02	CT	16.56		
Ours	sum-sum	5.07	sum-sum	**17.14**	sum-sum	**41.03**

(b) Slot F1 score on seen slots

Dataset	Snips		MultiWOZ 2.3		SGD	
Baselines	Coach	47.34	Coach	64.79	Coach	49.11
	RZT	**54.46**	CT	**73.36**		
Ours	sum-sum	53.33	sum-sum	68.13	sum-sum	**54.01**

the predictions of unseen slots and seen slots. Table 4.9 shows the averaged slot F1 score of all unseen slots and seen slots in each dataset. As seen in the table, for seen slots, our model significantly improved Coach by 5.99, 3.34, and 6.18 on Snips, MultiWOZ 2.3, and SGD, respectively. These results demonstrated that with the improvement in the slot entity prediction as the premise, our model improved zero-shot slot filling for seen slots. For unseen slots, our model improves Coach by 0.32 and 0.03 of the average slot F1 score on MultiWOZ 2.3 and SGD datasets, respectively. On the other hand, our model was defeated a little on the Snips dataset. The reason is that the model was simultaneously optimized for stage one and stage two. The performances were obtained from the model with the best overall performance on zero-shot slot filling. In practice, stage one and stage two were usually not able to be best optimized at the same time. Therefore, although our model improved the slot entity predictions in stage one, stage two in our model was not optimized as in Coach, so our model was defeated a little on some unseen slots.

By comparing our model with the best methods of RZT, CT, and Coach on Snips, MultiWOZ 2.3, and SGD dataset, one can see that our model outperformed best baselines on unseen slots of all datasets. On the other hand, our model did not outperform RZT and CT on seen slots of Snips and MultiWOZ 2.3 datasets. As discussed above, stage two of our model was the limitation to achieve better performance than CT on the MultiWOZ 2.3 dataset. The results on seen slots suggested that stage two of Coach was not as effective as the one-stage methods on handling seen slots. Therefore, considering to improve the robustness towards seen slots can be considered an approach to improve slot type predictions for two-stage methods.

Table 4.10: Comparison result of the multi-relation-based representation to IRPs on the Snips2017 dataset

Domain	IRP	Multi-relation-based
AddToPlaylist	**44.62**	40.78
BookRestaurant	34.62	**37.55**
GetWeather	**60.05**	51.76
PlayMusic	**39.42**	33.22
RateBook	20.28	**20.56**
SearchCreativeWork	74.29	**81.58**
SearchScreeningEvent	37.04	**38.01**
Average	**44.33**	43.35

Finally, we show the comparison result of the multi-relation-based representation and IRPs on the Snips2017 dataset in Table 4.10. As seen in the table, although IRP performed better than the multi-relation-based representation on the average performance, the multi-relation-based representation outperformed IRP over half of domains. This result demonstrated that the multi-relation-based representation and IRP have their own advantages on zero-shot slot filling.

4.5.2 Case studies

Finally, to intuitively show the improvements on slot entity predictions due to the proposed representation, we show two examples by our model compared to Coach in Figures 4.2 and 4.3.

Figure 4.2 shows the utterance 'Can you play a sonata.' from the Play Music domain from the Snips dataset. The correct IOB-slot tags indicate that 'sonata' is a slot entity and corresponding to the slot type 'genre.' In the experiments, when the Play Music is set to be the zero-shot target domain, the slot 'genre' was an unseen slot that was not appeared in training. The IOB tags of ours, Coach, and RZT are the slot entity prediction results of corresponding models. As seen in the figure, Coach and the best method of the Snips dataset, RZT, incorrectly predicted the 'sonata' as a slot entity. This is because, due to the domain shift problem, they hardly predicted the slot entities from the context corresponding to unseen slots. On the other hand, our model predicted the slot entity correctly. This case confirmed that the proposed representation is effective in dealing with the slot entities that correspond to unseen slots.

In Figure 4.3, the Rate Book domain is a seen domain in the experiment, the Search Creative Work domain is the unseen domain. The slot 'object

Zero-shot target domain: Play Music

Utterance	Can	you	play	a	sonata
Correct	O	O	O	O	B-genre
Ours	O	O	O	O	B
Coach	O	O	O	B	I
RZT	O	O	O	O	O

Figure 4.2: Example of slot entity prediction results on the slot entity corresponding to unseen slots

Training domain: Rate Book

Utterance	Give	This	novel	5	stars
Correct	O	B-object select	B-object type	B-rating value	I-rating value

Zero-shot target domain: Search Creative Work

Utterance	Find	a	movie	called	no	more	sadface
Correct	O	O	B-object type	O	B-object name	I-object name	I-object name
Ours	O	O	B	O	B	I	I
Coach	O	O	O	O	B	I	I
RZT	O	O	O	O	B	I	I

Figure 4.3: Example of slot entity prediction results on the slot entity corresponding to seen slots

type' in the Search Creative Work domain is explained differently from that in the Rate Book domain. Specifically, the 'object type' indicates a book type in the Rate Book domain, such as a novel or literature. While this slot indicates the type of creative work in the Search Creative Work domain, such as a movie or song. In Figure 4.3, the utterance and correct IOB-slot tags from the Rate Book domain show an instance that mentioned 'object type' in the Rate Book domain. The utterance and correct IOB-slot tags from the Search Creative Work domain show an instance that mentioned 'object type' in the Search Creative Work domain. In the utterance 'Find a movie called no more sadface,' the 'movie' is a slot entity that corresponds to the slot 'object type.' As seen in the figure, Coach and RZT missed predicting the 'movie' as a slot entity. This is because although the 'movie' corresponds to a seen slot type 'object type,' the context corresponding to the differently explained 'object type' was unseen in training. Coach only considering contextual information but not considering the semantic information of slots. RZT learned the slot entity patterns corresponding to specific slots but does not consider

the change of context environments. Therefore, they hardly handles the slot entities from the unseen context that correspond to differently explained seen slots. In contrast, our model using the proposed representation correctly predicted the 'movie' as a slot entity. This case confirmed that the proposed representation is more effective in predicting the slot entities from the context corresponding to differently explained seen slots.

4.6 Summary

In this chapter, we proposed the multi-relation-based representation for dealing with the domain shift problem of different context distributions on slot entity predictions. As discussed above, our model using the proposed multi-relation-based representation effectively alleviated domain shift problems for slot entity prediction and improved zero-shot slot filling. Note that we used simple structures on constructing models and did not use algorithms that can be expected to improve model performance further, such as attention mechanism. This is because our main goal was to evaluate the effectiveness of the proposed representation. Adding attention mechanism is considered as future work to further improve the model performance.

However, as shown in the results, the method of the previous study for slot type predictions was ineffective in dealing with the domain shift problem of unseen slots, so the improvement on slot entity predictions was not fully brought out and the performance on unseen slots was not improved much. Meanwhile, since the performance of complete zero-shot slot filling is the key towards the ultimate goal of the task-oriented dialogue system, only improving the slot entity prediction was not enough. Therefore, it is necessary to explore the representation that could alleviate the domain shift problems on slot type predictions.

Chapter 5

Ontology-based representation

5.1 Objective

To deal with the domain shift problems of unseen slots as well as differently explained seen slots on slot type predictions, we aimed to mine a representation that could fill the knowledge gap between the source domains and target domains. For this purpose, this chapter introduces the ontology-based representation that describes the relationships between slots and values. Specifically, we proposed two methods using ontology-based representations to deal with domain shift problems in zero-shot slot filling. This chapter first compared the two methods by preliminary experiments to choose a better one from section 5.1 to section 5.4. From section 5.5, we combine the better ontology-based representation with the multi-relation-based representation to further alleviate domain shift problems in zero-shot slot filling.

5.2 Ontology

As discussed in previous chapters, the knowledge of unseen slots and differently explained seen slots are hardly learned by the model. We moved our sights to the ontology to fill the knowledge gap between the source domains and target domains. In general, the basic mission of an ontology is to figure out what objects are, what ideas are, and how they relate to each other. In dialogue systems, the ontology usually represents a knowledge base that describes the relationships between slots and their values in a given dataset [42, 43, 88].

Figure 5.1 gives an example of a part of the ontology of the Get Weather domain from the Snips dataset. As seen in the figure, we could obtain all possible values for a given slot. For instance, given a slot 'country,' we can obtain all possible values as 'Belgium' and 'Canada' from the ontology. By reversing the use of the ontology, we could also obtain the slots that correspond to a given value. For instance, given a value 'storm,' we can

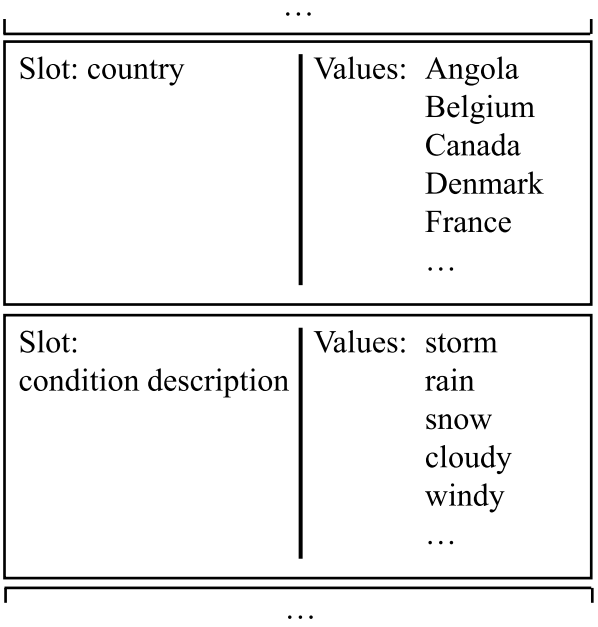

Figure 5.1: Example of relationships between slots and values described in the ontology

obtain all slots that have this value, which is the slot 'condition description' in this example.

5.3 Ontology-based complementary knowledge

In practice, an ontology defines all possible slots and all possible values of each slot for a given dataset. In this chapter, we assume the ontology for each domain, whether source domain or target domain, is fully-defined. Based on the fully-defined ontology, we use the definitions belonging to the source domains in the training process. In the test process, we use the definitions belonging to target domains in the ontology as the knowledge of target domains. Since the ontology is used as a knowledge base that is independent of any specific domain, the ontology does not change even if the domain changes from one to another. Thus, the definitions in the ontology become the representations that describe the intrinsic relationships between slots and their values across domains. We used the ontology for complementary knowledge to alleviate the problem of lacking knowledge in new domains for handling unseen slots and seen slots.

We proposed two two-stage methods to utilize the ontology in zero-shot slot filling. Specifically, stage one in both methods predicts slot entities, and

we used the ontology for slot type prediction in stage two.

We first proposed a two-stage method that predicts slot entities in stage one and predicts slot types in stage two by matching slot entities with the values in the ontology. Specifically, we used identical matching that judge whether two texts are totally the same or not. For brevity, we name this method 'ontology-matching.' In stage one, we used slot entity prediction methods, such as Coach's stage one, to predict IOB tags from the given utterance to identify slot entities. In stage two, based on the IOB tags, we extract the text sequence of each slot entity from the given utterance. Then, we tried to match the slot entity's text to the values in the ontology. For any value that matched the slot entity's text, the slot corresponding to the matched value was chosen as the slot type candidate. After attempting to match for all values in the ontology, we choose the slot type predictions based on the following three situations of the number of candidates: (1) If there is only one slot type candidate, the slot type candidate is chosen as the slot type prediction for the given slot entity; (2) If there are multiple slot type candidates, we randomly choose one candidate as the slot type prediction; (3) If there are no slot type candidates, the slot type prediction will not be performed, which means no slot type will be predicted as output. Meanwhile, the situation of no slot candidates means no values were matched to the given slot entity. This means the given slot entity should not be a value since this slot entity does not appear in the ontology containing all possible values. We treat the slot entity in this case as an incorrect prediction, and we correct the IOB tags of such a slot entity to 'O' tags to make the corresponding token chunk not to be a slot entity.

The ontology-matching method handles the slot type prediction using the relationships between slots and values in the ontology. The ontology complements the knowledge gap between the source domains and target domains to some extent. Therefore, this method could be expected to alleviate domain shift problems to handle unseen slots and seen slots in zero-shot slot filling. However, one problem should be considered in the situation (2) of the method. In this situation, we randomly chose a slot candidate as the prediction since there is no other context or slot information to choose the candidate. This could lead to incorrect slot type predictions.

Considering the problem mentioned above, we proposed another two-stage method in which we use the ontology for constraints on choosing the slot type candidates and predict slot types from candidates considering context and slot information. For brevity, we name this method 'ontology-constraints.' In this method, similar to the ontology-matching method, stage one predicts IOB tags to identify slot entities in the given utterance. In stage two, we tried to match the given slot entity to the values in the ontology to

obtain slot type candidates and perform the slot type prediction following the process of the 'ontology-matching' method, except the situation (2), which corresponds to multiple slot type candidates.

In the situation (2) of multiple slot type candidates, we first extracted the context vector of slot entities from stage one. Then we used an encoder such as BiLSTM to encode the context vectors into slot entity encoding representation. Next, we obtain the vector representations of each slot candidate by pooling the embedding representations of each slot description's tokens, which is the same process in obtaining the slot type vector representation as described in Chapter 4. After that, we computed the softmax similarity score between the slot entity encoding representation and each candidate's vector representation. The candidate with the highest score is then chosen as the prediction result. The computations of the similarity score and the slot type prediction are formulated as follows:

$$sim_j = e_{se} \cdot e_{slot_j}, slot_k \in S_c, \tag{5.1}$$

$$slot_{pred} = argmax(softmax(sim_1, sim_2, ..., sim_j, ...)). \tag{5.2}$$

here, S_c is the set of slot type candidates, e_{se} is the slot entity encoding representation, e_{slot_j} indicates the slot type representation of the $j-th$ candidate in S_c. The similarity score sim_j is the inner product of e_{se} and e_{slot_j}. The softmax function changes the values of similarity scores into probability distributions. The $slot_{pred}$ is the slot type prediction.

In this method, the ontology provides the relations between slots and their values as constraints to choose slot type candidates. The ontology-based representation can thus fill the knowledge gap between domains to some extent. Meanwhile, the slot type prediction from multiple candidates considers the context information based on the slot entity encodings and the slot type information based on the slot types. Thus, multiple slot candidates could be distinguished by the proposed method, and a suitable one could be chosen. Therefore, the proposed method can be expected to complement the lack of knowledge between domains and handle unseen slots and seen slots.

5.4 Preliminary experiments

5.4.1 Model construction

We constructed two two-stage models to use the proposed methods using ontology-based representations into zero-shot slot filling. To precisely evaluate the effectiveness of the ontology-based methods, we control the variable of the slot entity prediction. Specifically, we used the previous

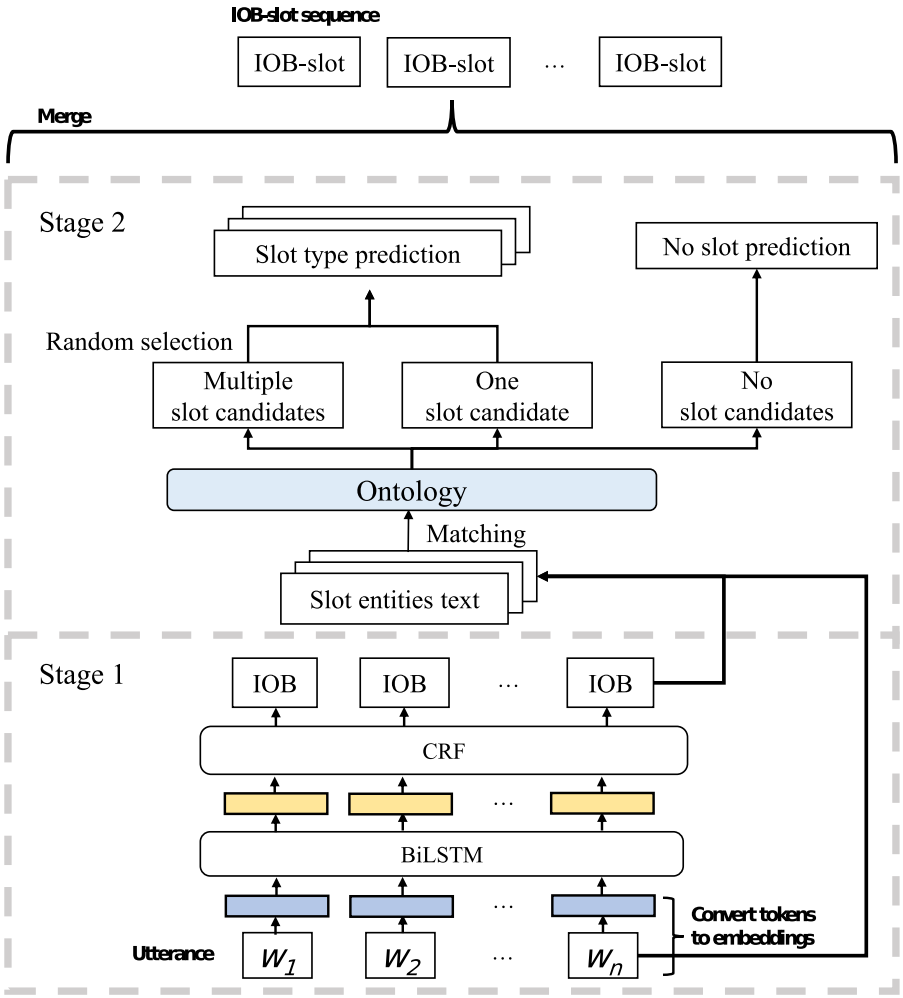

Figure 5.2: Two-stage model structure of using the ontology-matching method for slot type predictions in stage two

entity-context representation from Coach [22] as stage one of each model to handle slot entity predictions. Stage two of each model handles slot type prediction using the ontology-matching method and ontology-constraints method, respectively. Figure 5.2 and 5.3 shows the structure of the two models. For convenience, we name the two models 'ontology-matching model' and 'ontology-constraints model.' The slot description in the figure indicates the slot type vector representation, and the slot entity encoding in the figure indicates the slot entity encoding representation.

In stage one of both models, we first converted the tokens in the given utterance into embedding representations. Then, we used a BiLSTM layer

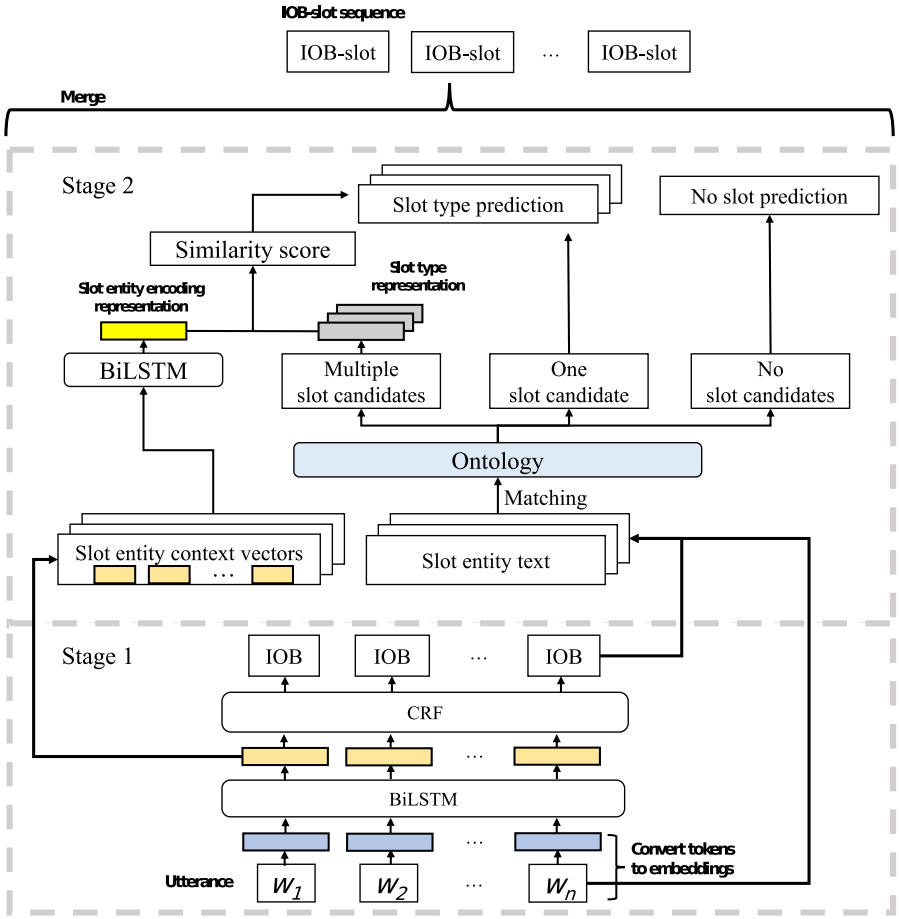

Figure 5.3: Two-stage model structure of using the ontology-constraint method for slot type predictions in stage two

to encode the tokens' embeddings into context encodings. The context encodings are then fed to the CRF layer to obtain the IOB digits on each token. The IOB tag with the highest probability is chosen as the prediction for each token.

As shown in Figure 5.2, in stage two of the ontology-matching model, we first identified the slot entities to extract the text sequence of each slot entity based on the IOB tags from stage one, following the same process described in Chapter 4. After obtaining the slot entities, we obtained the slot type candidates by matching the slot entities to the values in the ontology. Based on the situations of the slot type candidates, we chose the slot type prediction following the process described in section 5.2. As a consequence, we obtained slot type predictions for the slot entities that could match at

least one value, or we canceled the slot entities that matched no values in the ontology. Finally, the model merges the IOB predictions from stage one and the slot type predictions from stage two into an IOB-slot tag sequence as the final output.

As shown in Figure 5.3, In stage two of the ontology-constraints model, we first identified the slot entities to extract the text sequence of each slot entity in the same process as that in the ontology-matching model. Meanwhile, we extracted the context vector sequences corresponding to each slot entity. With the text sequences of slot entities, we obtained the slot type candidates in the process described in section 5.2. Then, we used a BiLSTM to encode the context vectors of each slot entity separately. As a consequence, we obtained the slot type candidates and slot entity encoding of each slot entity. Next, we choose the slot type prediction in the process described in section 5.2. Specifically, we obtained the slot type vector representations for the situation of multiple candidates. We used eq 5.1 and eq 5.2 to compute the softmax similarity score between the slot entity encoding and each candidate's vector representation. Then, we chose the slot type with the highest score as the slot type prediction. Finally, the model merges the IOB predictions from stage one and the slot type predictions from stage two into an IOB-slot tag sequence as the final output.

5.4.2 Experiment setting

For preliminarily evaluating the effectiveness of the two ontology-based methods, we conducted our experiments on Snips and MultiWOZ 2.3 datasets. As described in previous chapters, the Snips dataset contains seven domains with 14484 utterances, 28 single-domain slots, and 25 multi-domain slots. The MultiWOZ 2.3 dataset contains six domains with 43591 utterances, six single-domain slots, and 24 multi-domain slots after the data cleaning. We built ontologies for Snips and MultiWOZ 2.3 datasets by extracting all slots and their values from the annotations of the datasets. The ontology of the Snips dataset contains 53 slots with 245 values for each slot in average. The ontology of the MultiWOZ 2.3 dataset contains 30 slots with 80 values for each slot in average. The numbers of the values for different slots were diverse. The slots describing names or times contains much more values than other slots.

We used the two-stage methods Coach [22] and PCLC [23] as baseline methods. Since stage one of these two baselines is the same as used in our models, the evaluation could reflect the effectiveness of stage two of our models more precisely. In the preliminary experiment, we aimed to find a better method to use the ontology-based representation, so we did not

compare with the best method for each dataset.

For the slot entity and slot type representations, we used the sum-slot and sum-enc representations that are the same as in Coach to control variables. For the parameter settings, we set the BiLSTM in stage one as one layer with 256 hidden units. In stage two of the ontology-matching model, no learning parameters need to be set. In stage two of the ontology-constraints model, we set the entity encoder BiLSTM as two layers with 256 hidden units in each layer. For the word representations, loss functions, and optimizer, we used the same setting as the experiment in the last chapter. For the training strategies, evaluation metrics, and the settings for training, validation, and test sets, we also used the same settings as the experiments in the last chapter to make fair comparisons.

We used the slot F1 score for evaluating the model performance on each domain, and we used the slot F1 score on each slot to evaluate the model performance on unseen and seen slots. One-way ANOVA was used to measure the significance level of the difference between methods.

5.5 Preliminary results and discussions

Tables 5.1 and 5.2 show the slot F1 scores of different methods on Snips and MultiWOZ 2.3 datasets, respectively. In the tables, domain indicates the model performance on each zero-shot target domain. Average means the average performance among all domains in a dataset. The underlined numbers indicate the better baseline method, and the bold numbers are the best results on each domain and on average. The OM and OC indicate the ontology-matching model and the ontology-constraints model, respectively. The '***' indicates the corresponding proposed model has a significant difference compared to the better baseline method on the level of $p < 0.001$.

As seen in Tables 5.1 and 5.2, Coach is the better baseline method on almost all domains and average performances compared to PCLC, achieving 33.42 and 59.55 in the average slot F1 scores on Snips and MultiWOZ 2.3 datasets, respectively. By comparing the OM model and Coach, one can see that the OM model outperformed Coach by 27.07 and 1.17 on Snips and MultiWOZ 2.3 datasets, respectively. By comparing the OC model and Coach, one can see that the OC model outperformed Coach on Snips and MultiWOZ 2.3 datasets by 28.22 and 8.12, respectively. These results demonstrated that by using the relationships between slots and values, the ontology-based methods generally improved zero-shot slot filling.

Table 5.3 shows the averaged slot F1 scores of different methods on unseen and seen slots in Snips and MultiWOZ 2.3 datasets. Table 5.3 (a) shows the

Table 5.1: Results by different methods on the Snips dataset

	Baselines		Ours	
Domain	Coach	PCLC	OM	OC
AddToPlaylist	47.30	46.92	68.93	**71.11**
BookRestaurant	31.30	27.25	**67.89**	67.59
GetWeather	45.97	37.92	**75.23**	72.01
PlayMusic	29.99	20.11	60.94	**63.88**
RateBook	9.60	20.08	38.86	**42.54**
SearchCreativeWork	50.55	49.21	64.37	**65.95**
SearchScreeningEvent	19.19	14.99	47.18	**48.41**
Average	33.42	30.93	60.49***	**61.64***

Table 5.2: Results by different methods on the MultiWOZ 2.3 dataset

	Baselines		Ours	
Domain	Coach	PCLC	OM	OC
Attraction	66.07	68.49	**69.79**	67.43
Hotel	64.24	62.16	61.34	**67.23**
Restaurant	70.40	68.73	75.81	**81.28**
Taxi	39.17	38.30	32.64	**49.04**
Train	62.66	48.63	54.12	**67.99**
Others	54.75	55.77	70.60	**73.02**
Average	59.55	57.01	60.72	**67.67***

results on unseen slots, and (b) shows the results on seen slots. The '**' and '***' indicate that the corresponding proposed model has a significant difference compared to the better baseline method on the level of $p < 0.01$, and $p < 0.001$. As seen in the table, Coach is a better baseline on unseen and seen slots for both datasets. By comparing our models' performances with baselines, one can see that the OM model outperformed Coach by 42.01 and 12.28 on Snips and MultiWOZ 2.3 datasets for unseen slots, respectively. The OC model further outperformed Coach by 44.69 and 20.96 on Snips and MultiWOZ 2.3 datasets, respectively. While for the seen slots, the OM model outperformed Coach by 22.19 on the Snips dataset but defeated by Coach by 0.89 on the MultiWOZ 2.3 dataset. The reason will be discussed in the following parts. On the other hand, the OC model outperformed Coach by 20.79 and 6.06 on Snips and MultiWOZ 2.3 datasets, respectively.

By comparing OM and OC models, the OC model achieved better performance than the OM model on the unseen slots for both datasets and the seen slots for the MultiWOZ 2.3 dataset. While the OM model outperformed

Table 5.3: Results by different methods on unseen and seen slots

(a) Slot F1 score on unseen slots

Dataset	Snips		MultiWOZ 2.3	
Baselines	Coach	5.23	Coach	16.82
	PCLC	5.01	PCLC	16.47
Ours	OM	47.24	OM	29.10**
	OC	**49.92**	OC	**37.78****

(b) Slot F1 score on seen slots

Dataset	Snips		MultiWOZ 2.3	
Baselines	Coach	47.34	Coach	64.79
	PCLC	39.35	PCLC	63.03
Ours	OM	**69.53*****	OM	63.90
	OC	68.13***	OC	**70.85****

the OC model a bit on the seen slots for the Snips dataset. The main reason was that some slots in one domain had shared values, and the OM method could not judge the correct slot for a given value by random choosing. For instance, in the Taxi domain from the MultiWOZ 2.3 dataset, the slots 'arrive at' and 'leave by' have shared values of time, such as 11:00, 12:00. The OM method could not judge which slot a given time should belong to. On the other hand, the OC method took context information into account to choose the correct slot, so the OC method is more robust in dealing with the values shared by different slots. In addition, the MultiWOZ 2.3 dataset contains many slots that have shared values. Besides the 'arrive by' and 'leave at' introduced above, the 'book people' and 'stars' in the Hotel domain have shared values of numbers, and the 'departure' and 'destination' in the Train domain have shared values of locations. Compared to the MultiWOZ 2.3 dataset, the Snips dataset has much fewer slots that have shared values. This is also the reason that the OM method achieved much lower performance than the OC method and even lower performance than the baseline on seen slots on the MultiWOZ 2.3 dataset.

In summary, compared to the OM's limited advantage on seen slots of the Snips dataset, the insufficient of distinguishing the slots of shared values could be considered a critical miss on zero-shot slot filling in practice. Therefore, the results demonstrated that by choosing the slot type with the consideration of context and slot information, the OC method deals with the slot type prediction better than the OM method. With this, we combined the method using multi-relation-representation and the ontology-constraint method to further alleviate the domain shift problems in zero-shot slot filling.

5.6 Experiment of combined method

5.6.1 Model construction

To further alleviate domain shift problems in zero-shot slot filling, we combined the multi-relation-based representation and the ontology-based representation to construct a two-stage model. Figure 5.4 shows the model structure. Stage one follows the process of using the multi-relation-based method described in section 4.3 in Chapter 4. Specifically, we obtained the IOB tag predictions from stage one, which indicates the slot entities in the given utterance. Stage two of the model follows the process of the ontology-constraint model described in section 5.3. Specifically, we obtained slot type predictions from stage two. Finally, the model merges the IOB predictions from stage one and slot type predictions from stage two into an IOB-slot tag sequence as the final output.

In this part, we compared the effectiveness of the mean-slot and sum-slot slot type vector representations and the effectiveness of the last-enc and sum-enc slot entity encoding representations.

5.6.2 Experiment setting

To comprehensively evaluate the proposed method using the multi-relation-based and the ontology-based representations, we conducted experiments on the Snips, MultiWOZ 2.3, and SGD datasets. As described in previous chapters, the Snips dataset contains seven domains with 14484 utterances, 28 single-domain slots, and 25 multi-domain slots. The MultiWOZ 2.3 dataset contains six domains with 43591 utterances, six single-domain slots, and 24 multi-domain slots after the data cleaning. The SGD dataset contains 20 domains with 70166 utterances, 63 single-domain slots, and 38 multi-domain slots after data cleaning. In addition, we also conducted the experiment on the Snips2017 dataset to compare the performance with IRPs that introduced in the Chapter 3.

We follow the same process to build the ontology for SGD to that for Snips and MultiWOZ 2.3 datasets. Specifically, we built the ontology for the SGD dataset by extracting all slots and their values from the annotations of the dataset. The ontology of the SGD dataset contains 101 slots with 109 values for each slot in average. Similar to Snips and MultiWOZ 2.3 datasets, for the SGD dataset, the slots describing names or times contains much more values than other slots.

To comprehensively evaluate our methods, we used the methods from one-stage methods and two-stage methods including the best method on each

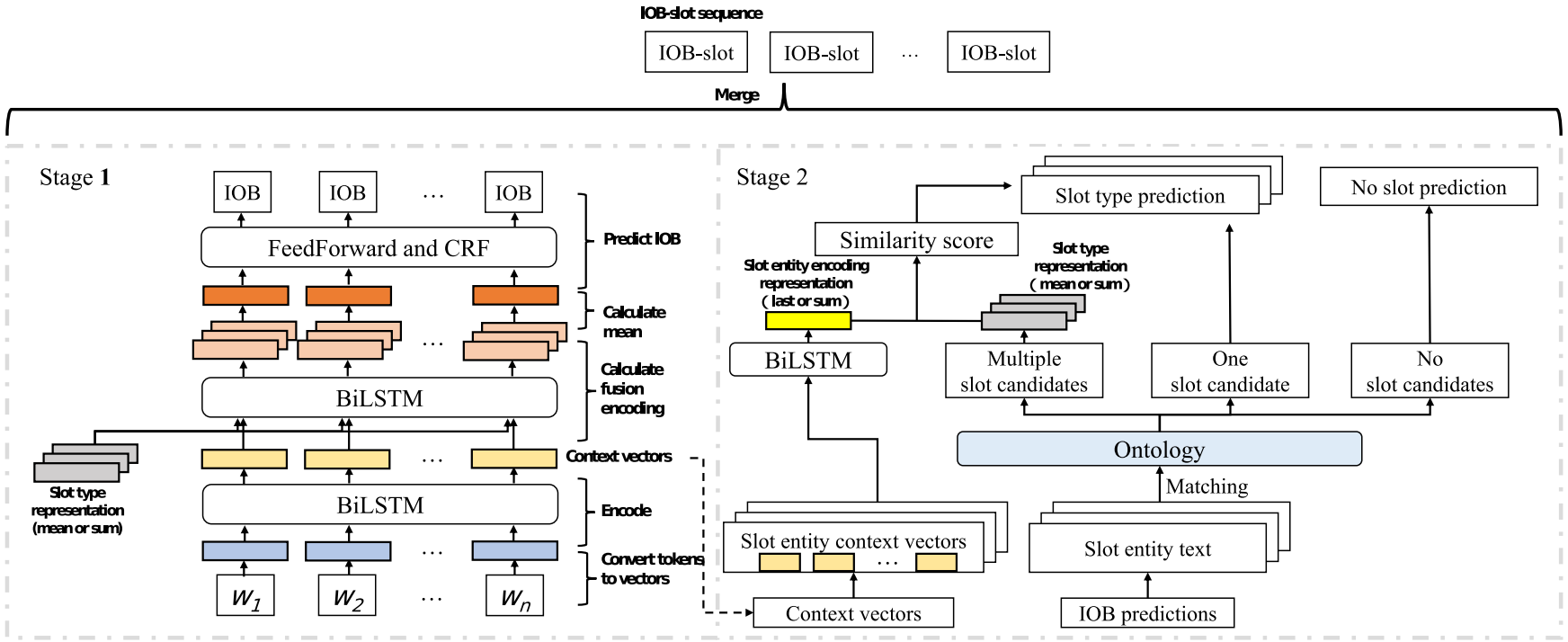

Figure 5.4: Model structure of the the combination use of intrinsic representations

dataset. Specifically, we used CT [19] (best on the MultiWOZ 2.3 dataset), RZT [21] (best on the Snips dataset), Coach [22] (best on the SGD dataset), and PCLC [23] as baselines. Note that we did not use ZAT [20] as a baseline since ZAT has a huge time-consuming when training on the MultiWOZ 2.3 dataset and has an unacceptable time-consuming when training on the SGD dataset.

For parameter settings, we used the same parameters described in section 4.3 in Chapter 4 for stage one. We used the same parameters for stage two as the ontology-constraint model described in section 5.3. For the training strategies, evaluation metrics, and the settings for training, validation, and test sets, we followed the same settings as the experiments in previous chapters to make fair comparisons.

In addition, we also conducted one-shot and few-shot learning experiments on Snips and MultiWOZ 2.3 datasets to evaluate the robustness of our models. Specifically, we use on instance from the zero-shot target domain for training in the one-shot setting. And we use 25% and 50% instances from the target domain in few-shot setting. To guarantee the reproducibility, we use the one, 25%, and 50% instances from the beginning of each domain's index. According to the one-shot and few-shot settings, we cannot split the same validation set from the zero-shot target domain as our previous experiments did, as the previous validation set used the first 500 instances from the target domain that overlapped with few-shot training instances. Therefore, we split the validation set from source domains and use this validation set to choose the best models. Specifically, we use first 80% instances of each source domain and all data of few-shot instances from the target domain for training, and we use the remained 20% instances from each source domains for validation. The remaining data from the target domain was used for test. To make a fair comparison among zero-shot setting, one-shot and few-shot settings, we also conducted zero-shot slot filling in the similar setting by using the first 80% instances of each source domain for training the last 20% instances for validation. In this setting, all instances from zero-shot target domains were used for test.

5.7 Results and discussions

5.7.1 Experimental results and discussions

Tables 5.4, 5.5, and 5.6 list the slot F1 score results for zero-shot slot filling for Snips, MultiWOZ 2.3, and SGD datasets, respectively. Limited by the format of the dissertation format, we could not show the methods' results

Table 5.4: Results on the Snips dataset

(a) Results of Baseline methods				
Domain	CT	RZT	Coach	PCLC
AddToPlaylist	46.03	<u>50.27</u>	47.30	46.92
BookRestaurant	<u>39.26</u>	34.52	31.30	27.25
GetWeather	<u>66.72</u>	61.44	45.97	37.92
PlayMusic	39.33	<u>39.70</u>	29.99	20.11
RateBook	13.83	<u>25.01</u>	9.60	20.08
SearchCreativeWork	36.65	38.40	<u>50.55</u>	49.21
SearchScreeningEvent	<u>23.76</u>	20.15	19.19	14.99
Average	37.94	<u>38.50</u>	33.42	30.93
(b) Results of proposed methods				
Domain	mean-last	mean-sum	sum-last	sum-sum
AddToPlaylist	70.74	**71.49**	71.22	71.02
BookRestaurant	70.69	**73.57**	71.52	72.02
GetWeather	82.94	**83.80**	80.42	83.55
PlayMusic	78.84	**80.32**	78.49	78.88
RateBook	40.37	**43.66**	38.84	38.41
SearchCreativeWork	65.01	64.27	65.17	**68.07**
SearchScreeningEvent	57.85	**59.69**	57.87	58.81
Average	66.63***	**68.12***	66.22***	67.25***

for one domain in a single row. Accordingly, we split each table into two parts, in which (a) lists the results from baseline methods and (b) lists the results from the proposed models. In the tables, underlined numbers indicate the best baselines for each dataset. Bold numbers indicate the best results among the methods in each domain. The results from the proposed model were obtained with four combinations of slot type vector representations and slot entity encoding representations. The '*,' '**,' and '***' indicate that the corresponding proposed model has a significant difference compared to the best baseline method on the level of $p < 0.05$, $p < 0.01$, and $p < 0.001$.

By comparing the results for our models in Tables 5.4, 5.5, and 5.6, one can see that the mean-sum combination demonstrated the best performance on the Snips and SGD datasets, and the sum-sum combination demonstrated the best performance on the MultiWOZ 2.3 dataset. For the Snips dataset, RZT was the best baseline method. Our model outperformed RZT in all domains. The best-preforming mean-sum combination outperformed RZT by 29.62 of the average slot F1 score. For the SGD dataset, Coach was the best baseline method. In this case, our models outperformed Coach in almost all

Table 5.5: Results on the MultiWOZ 2.3 dataset

(a) Results of Baseline methods

Domain	CT	RZT	Coach	PCLC
Attraction	65.94	64.86	66.07	68.49
Hotel	66.89	54.32	64.24	62.16
Restaurant	71.06	58.74	70.40	68.73
Taxi	**52.86**	51.16	39.17	38.30
Train	80.19	63.03	62.66	48.63
Others	58.49	58.93	54.75	55.77
Average	65.91	58.51	59.55	57.01

(b) Results of the proposed methods

Domain	mean-last	mean-sum	sum-last	sum-sum
Attraction	70.28	71.15	71.34	**75.39**
Hotel	66.62	**67.85**	66.82	66.93
Restaurant	**78.10**	77.28	77.43	76.57
Taxi	45.81	50.09	49.85	51.27
Train	72.82	**75.09**	71.21	74.78
Others	73.23	73.14	70.99	**73.86**
Average	67.81	69.10*	67.94	**69.80***

domains. The best-performing mean-sum combination outperformed Coach by 10.38 of the average F1 score. Finally, for the MultiWOZ 2.3 dataset, CT was the best of the baseline methods. Our models outperformed CT in almost all domains. The best-performing sum-sum combination outperformed CT by 3.89 of the average slot F1 score. These results demonstrated that by further alleviating domain shift problems, the use of the multi-relation-based and ontology-based intrinsic representations significantly improved the performance of zero-shot slot filling. Note that the results of PCLC are not consistent with the results in our publication [1]. This is because we employed the original implementation of PCLC in the publication [1]. The original implementation used modified slot name descriptions and the incorrect optimization process, which were introduced in Chapter 2. We used the universal slot name descriptions and correct optimization process for PCLC to obtain the results in this chapter so that the results were different.

The results showed that our models improved baselines more significantly on the Snips dataset than on the other two datasets. According to the slot distributions in each domain described in Chapter 2, we speculate the reason is that in the Snips dataset, the slots and contexts in the target domains are more unrelated to the slots and contexts in source domains than that in the other two datasets. For instance, for the target domain Book

Table 5.6: Results on the SGD dataset

(a) Results of Baseline methods				
Domain	CT	RZT	Coach	PCLC
Alarm	77.72	70.13	85.72	84.57
Banks	26.31	21.93	58.37	9.03
Buses	37.38	35.43	34.47	34.23
Calendar	77.56	77.80	81.51	77.52
Events	65.50	59.08	64.24	64.98
Flights	36.57	37.59	**44.21**	42.68
Homes	82.28	92.76	95.48	73.74
Hotels	39.57	46.68	44.86	40.82
Media	53.31	50.14	51.13	46.18
Messaging	53.51	49.57	73.50	59.94
Movies	55.50	59.19	64.93	60.56
Music	36.63	23.68	31.52	37.82
Payment	42.11	40.87	82.33	71.64
RentalCars	43.60	49.32	37.79	**50.56**
Restaurants	62.15	60.06	59.21	58.80
RideSharing	31.33	9.58	81.45	78.85
Services	91.95	91.39	95.36	90.47
Trains	52.56	56.08	91.11	87.08
Travel	94.75	94.07	98.45	98.26
Weather	95.76	95.26	98.66	98.61
Average	57.80	56.03	68.72	63.32

(b) Results of proposed methods				
Domain	mean-last	mean-sum	sum-last	sum-sum
Alarm	**86.64**	84.02	81.04	82.83
Banks	83.66	84.48	**84.91**	83.01
Buses	36.62	36.75	38.71	**38.84**
Calendar	86.93	86.99	86.79	**87.16**
Events	75.11	75.17	**75.91**	75.61
Flights	42.39	44.13	43.00	40.91
Homes	**98.98**	98.86	98.83	98.75
Hotels	47.53	**57.56**	50.37	44.92
Media	**75.64**	75.18	73.30	74.07
Messaging	91.42	91.95	**93.56**	92.47
Movies	93.93	**94.22**	93.61	93.48
Music	66.19	65.07	67.50	**67.85**
Payment	96.60	**97.81**	96.81	97.08
RentalCars	40.60	40.12	41.34	41.66

Restaurants	**74.90**	73.98	73.64	73.94
RideSharing	85.37	84.82	88.15	**88.91**
Services	99.34	**99.40**	99.33	99.33
Trains	93.82	94.03	93.61	**94.44**
Travel	**99.88**	99.82	99.66	99.76
Weather	**99.25**	99.19	99.22	99.23
Average	78.74***	**79.18***	78.96***	78.71***

Restaurant from the Snips dataset, the models were required to deal with unseen slots, including 'restaurant name' and 'served dish' that is unrelated to the training slots, such as 'album' and 'time range.' On the other hand, for the target domain 'Homes' in the SGD dataset, the models were required to deal with unseen slots, including 'area' and 'visit date' that are similar to some of the training slots, such as 'location' and 'date.' While in the MultiWOZ 2.3 dataset, the unseen slots were much fewer than that in Snips and SGD datasets. Based on such slot distributions, baseline models without using the knowledge of target domains were ineffective in dealing with those unseen slots and unseen contexts in the Snips dataset, while our model using intrinsic representations complemented the knowledge gap between domains. Therefore, our models improved baseline methods more significantly on the Snips dataset than on the other two datasets.

To identify the most effective combination of slot type vector representations and slot entity encoding representations, we performed a one-way ANOVA among four combinations on each dataset. Similar to the experiments in previous experiments, the one-ANOVA showed no significant difference among four combinations ($p > 0.1$) in each dataset. Therefore, we chose the most effective one by comparing the average performance of our models over three datasets. According to the results in Tables 5.4, 5.5, and 5.6, the average performance of mean-last, mean-sum, sum-last, and sum-sum combinations are 71.06, 72.13, 71.04, and 71.92, respectively. Accordingly, the mean-sum combination outperformed others on the averaged performance over three datasets, so it is the most effective combination to some extent.

According to the comparison results on the combinations of slot type vector representations and slot entity encoding representations so far, we could summarize that the sum-enc slot entity encoding representation is generally more effective than the last-enc representation in zero-shot slot filling. On the other hand, the mean-slot and sum-slot slot type vector representations each has their own advantage in different situations, but they do not significantly differ from each other. Therefore, the comparison results suggested that it is better to use the sum-enc representation to obtain the slot entity encodings, while the uses of mean-slot and sum-slot depend on

Table 5.7: Comparison result of the combined model to IRPs on the Snips2017 dataset

Domain	IRP	Combined model
AddToPlaylist	44.62	**72.89**
BookRestaurant	34.62	**76.28**
GetWeather	60.05	**80.64**
PlayMusic	39.42	**81.23**
RateBook	20.28	**51.30**
SearchCreativeWork	74.29	**86.54**
SearchScreeningEvent	37.04	**60.91**
Average	44.33	**72.83*** **

situations.

Following the results, we chose the model of mean-sum combination as our representative model and used it for the subsequential analysis.

Then, we show the comparison result of the combined model and IRPs on the Snips2017 dataset in Table 5.7. As seen in the table, the combined model outperfomed IRP on all domains and the average performance. This result demonstrated that by further alleviate domain shift problems, the combination of multi-relation-based and ontology-based representations are more effecitve than IRPs for zero-shot slot filling.

5.7.2 Results on unseen and seen slots

Next, we investigated the effectiveness of our model on different slot types. We compared the performance of our representative model and baselines for both unseen and seen slot types. Similar to our previous experiments, we computed the average slot F1 scores in a given dataset over all unseen slots and seen slots to compress the domain-dependent problem on performances. Table 5.8 shows the results. The bold and underlined numbers indicate each dataset's best results and the best baseline methods, respectively. The '*,' '**,' and '***' indicate that the corresponding proposed model has a significant difference compared to the best baseline method on the level of $p < 0.05$, $p < 0.01$, and $p < 0.001$.

As seen in Table 5.8 (a), for the prediction of unseen slots, Coach was the best baseline method for the Snips and SGD datasets, and RZT was the best baseline method for the MultiWOZ 2.3 dataset. In terms of the average slot F1 score, our model outperformed the best baselines of Coach on Snips and SGD, RZT on MultiWOZ 2.3 by 52.54, 19.47, and 9.49, respectively. These results demonstrated that the proposed intrinsic representations significantly

Table 5.8: Results on unseen and seen slots

(a) Slot F1 score on unseen slots

Dataset	Snips		MultiWOZ 2.3		SGD	
Baselines	CT	3.99	CT	16.56	CT	37.55
	RZT	5.02	RZT	19.95	RZT	38.44
	Coach	5.23	Coach	16.82	Coach	41.00
	PCLC	5.01	PCLC	16.47	PCLC	29.21
Ours	mean-sum	**57.77*****	mean-sum	**29.44***	mean-sum	**60.47*****

(b) Slot F1 score on seen slots

Dataset	Snips		MultiWOZ 2.3		SGD	
Baselines	CT	56.60	CT	73.36	CT	45.82
	RZT	54.46	RZT	58.94	RZT	40.65
	Coach	47.34	Coach	64.79	Coach	49.11
	PCLC	39.35	PCLC	63.03	PCLC	46.35
Ours	mean-sum	**74.63*****	mean-sum	**74.12**	mean-sum	**66.08****

improved zero-shot slot filling on unseen slot types.

As seen in Table 5.8 (b), CT was the best baseline method for the Snips and MultiWOZ 2.3 datasets, and Coach was the best baseline method for the SGD dataset. In terms of slot F1 score, our model outperformed the best baselines on Snips, MultiWOZ 2.3, and SGD datasets by 18.03, 0.76, and 16.97, respectively. These results demonstrated that the proposed intrinsic representations effectively handled seen slot types by the complement knowledge provided via the ontology.

5.7.3 Results on one-shot and few-shot settings

Then, we investigated the robustness of our model by one-shot and few-shot experiments. Figure 5.5 shows the comparison results between our representative model and the best methods on Snips and MultiWOZ 2.3 datasets. Note that as described in section 5.6.2, we used a different experiment setting for obtaining zero-shot slot filling performances for comparing with one-shot and few-shot results, so the zero-shot results in this section is different to that in previous sections. As seen in the figure, as training instances increase, the performance of our model and baseline models all increased. Our model outperformed baselines on every setting from one-shot to the few-shot of 50% training instances. These results demonstrated that our model is robust on different data situations by alleviating domain shift problems.

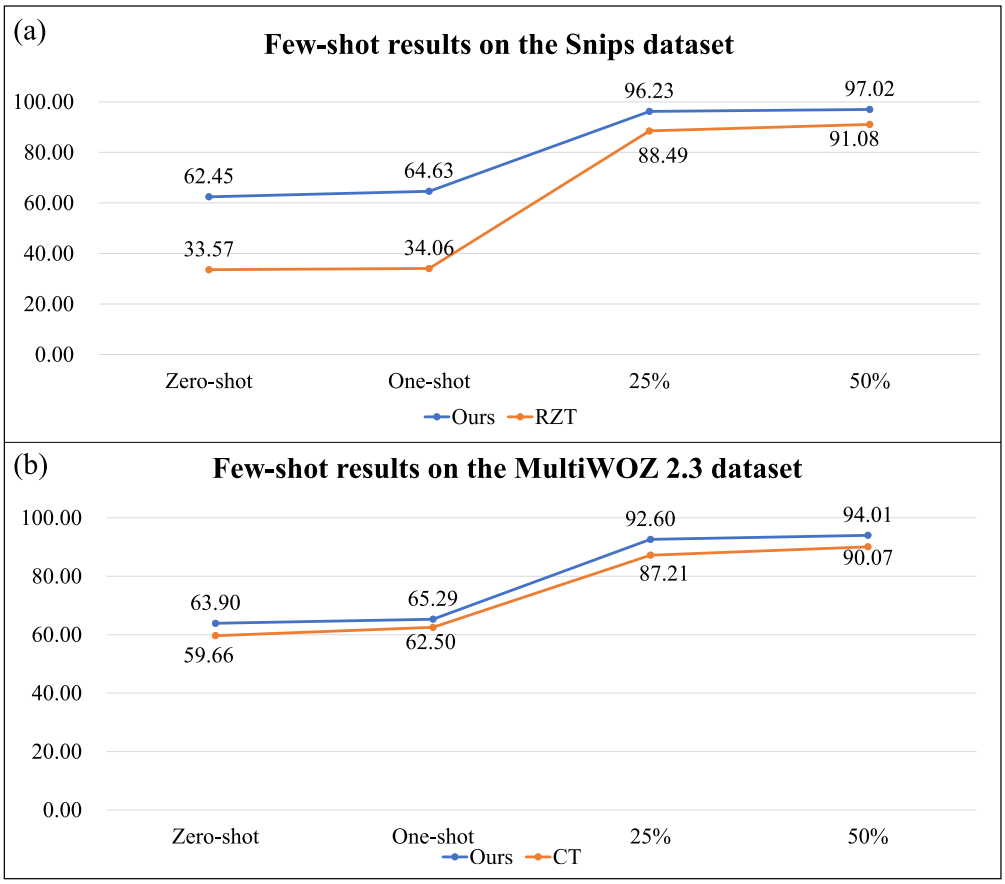

Figure 5.5: Comparison results between our model and the best method on each dataset in one-shot and few-shot settings. (a) shows the results on the Snips dataset; (b) shows the results on the MultiWOZ 2.3 dataset

5.7.4 Case studies

Finally, to intuitively show the improvement due to our model, we show two examples of the zero-shot slot filling results by our representative model compared with baseline methods in Figures 5.6 and 5.7.

Figure 5.6 shows the utterance 'Book Tun Tavern for morning in Norfolk Island' from the Book Restaurant domain in the Snips dataset. The correct tags indicate that 'Tun Tavern', 'morning', and 'Norfolk Island' are slot entities, and they correspond to the slot types 'restaurant name', 'time range', and 'country', respectively. In the experiments, the 'restaurant name' was an unseen slot type, while 'time range' and 'country' were seen slot types. As seen in the figure, the one-stage methods CT and RZT failed to capture the

Zero-shot target domain: BookRestaurant

Utterance	Book	Tun	Tavern	for	morning	in	Norfolk	Island
Correct	O	B- restaurant_name	I- restaurant_name	O	B- time_range	O	B- country	I- country
Ours	O	B- restaurant_name	I- restaurant_name	O	B- time_range	O	B- country	I- country
PCLC	O	B- city	I- city	O	B- time_range	O	B- country	I- country
Coach	B- restaurant_type	B- cuisine	I- cuisine	O	B- time_range	O	B- city	I- city
RZT	O	B- restaurant_name	O	O	B- time_range	O	B- country	I- country
CT	O	O	O	O	B- time_range	O	B- country	I- country

Figure 5.6: Example of zero-shot slot filling results showing improvement in the prediction of unseen slot

entire slot entity of 'Tun Tavern', which is consistent with the argument in [22]. For the 'Tun Tavern' slot entity, Coach and PCLC incorrectly predicted the slot type as 'cuisine' and 'city' because of the lacking knowledge in the Book Restaurant domain due to domain shift problems. Coach also incorrectly predicted 'book' as a slot entity. In contrast, our representative model correctly predicted each slot entity and each slot type.

In Figure 5.7, the slot type 'object name' was a seen slot type that appeared in the training domain Rate Book but has different explanations due to the contexts in the zero-shot target domain Search Creative Work. Specifically, the utterance and the correct tags of the Rate Book domain indicate that the 'object name' corresponds to 'The Forest', which is a book to be rated. On the other hand, the utterance and the correct tags of the Search Creative Work domain indicate that the 'object name' corresponds to a requested song called 'Bliss Torn from Emptiness'. As seen in the figure, the baseline methods CT, RZT, Coach, and PCLC all failed to predict the complete slot entity 'Bliss Torn from Emptiness' thus failed to predict the slot type 'object name'. In contrast, by capturing the general and specific characteristics of slot entities, our model using the multi-relation-based representation predicted the slot entity 'Bliss Torn from Emptiness' correctly. Then, by establishing the relationship between slots and values by the ontology-based representation, our model correctly predicted the slot type 'object name' for the slot entity.

In summary, these examples confirmed the effectiveness of the proposed intrinsic representations, and the combined use of the representations further alleviated the domain shift problems on the slot entity predictions from unseen contexts and the slot type predictions for unseen and seen slots.

Training domain: RateBook

Utterance	The	Forest	Should	be	rated	a	four
Correct	B- object_name	I- object_name	O	O	O	O	B- rating_value

Zero-shot target domain: SearchCreativeWork

Utterance	Play	Bliss	Torn	from	Emptiness
Correct	O	B- object_name	I- object_name	I- object_name	I- object_name
Ours	O	B- object_name	I- object_name	I- object_name	I- object_name
PCLC	O	B- object_name	I- object_name	O	B- object_name
Coach	O	B- object_name	I- object_name	O	B- object_name
RZT	O	B- object_name	I- object_name	O	B- object_name
CT	O	B- object_name	I- object_name	O	O

Figure 5.7: Example of zero-shot slot filling results showing improvement in the prediction of differently explained seen slot

5.8 Summary

In this chapter, we proposed ontology-based representations for dealing with domain shift problems of unseen slots and differently explained seen slots. As discussed above, by establishing the relationships between slots and values, the proposed representations effectively alleviated the domain shift problems and significantly improved zero-shot slot filling by improving slot type predictions. By combining the advantages the ontology-based representation and the multi-relation-based representation, the combination of intrinsic representations further alleviated domain shift problems and improved zero-shot slot filling.

In summary, towards the ultimate goal of the task-oriented dialogue system, the proposed representations generally alleviated domain shift problems in zero-shot slot filling. However, how the improvements on zero-shot slot filling contributes to the dialogue system was still not clear. Evaluating the improvements on zero-shot slot filling on the dialogue system level is necessary.

Chapter 6

Zero-shot slot filling's contribution on the task-oriented dialogue system

6.1 Objective

In this research, we have mined intrinsic representations to improve zero-shot slot filling. However, (1) the process of subsequential modules may alleviate the influence of the errors from previous modules. Zero-shot slot filling may not improve the performance of the entire dialogue system as much as slot filling compared to conventional methods. In previous studies, whether zero-shot slot filling benefits a complete dialogue system when encountering unseen domains remained unclear. (2) Furthermore, previous studies mainly focused on improving the overall performance of zero-shot slot filling, but they may improved the performance on different slots. These improvements may have different effects on the dialogue system. It was unclear how the improvements of zero-shot slot filling affects the performance of the entire dialogue system.

Clarifying these problems can provide suggestions for improving zero-shot slot filling. Therefore, we investigated how zero-shot slot filling affected and benefited the performance of the task-oriented dialogue system in this study. To do so, we constructed and compared dialogue systems using different slot filling modules when encountering unseen domains. Specifically, to make a precise investigation on the slot filling module, we built a base system and fixed the modules except the slot filling. Then, we replaced the slot filling module with models trained on zero-shot and conventional methods. Moreover, we employed an ideal slot filling process that could perfectly perform slot filling for comparison.

6.2 Dialogue system construction

6.2.1 ConvLab-2 platform

We employed the ConvLab-2 [14] platform to construct the dialogue system. ConvLab-2 is a toolkit for building, evaluating, and diagnosing dialogue systems. This toolkit has various pre-trained or pre-defined modules for quickly constructing a task-oriented dialogue system. Those prepared modules were trained or defined by adapting specific datasets, such as the MultiWOZ dataset [42]. Besides, ConvLab-2 provides an analyzer for diagnosing errors and mistakes in the dialogue systems. To conduct dialogue experiments, ConvLab-2 also provides a simulation agency to act as users to test the dialogue system.

ConvLab-2 supports the pipeline, semi-end-to-end, and end-to-end structures by using specific modules. For instance, ConvLab-2 can be used to build a pipeline system with the NLU of BERT-NLU, DST of rule-based DST, DP of rule-based policy or MLEPolicy, and NLG of Template NLG. ConvLab-2 can also be used to build a semi-end-to-end system with the NLU+DST of TRADE or SUMBT, the DP module, and the NLG module.

6.2.2 System construction

To reduce the influence of the accuracies of other modules on the performance of the dialogue system, we used the pipeline system that showed high performance as the base system: BERT-NLU + rule-based dialogue state tracking + rule-based policy + template-based natural language generation [18]. Each part of the base system can be directly employed with pre-trained or pre-defined modules from ConvLab-2, which were adapted to specific datasets.

To precisely investigate the contribution of zero-shot methods in dialogue systems when encountering unseen domains, we must guarantee the accuracies of other modules on unseen domains. Since we cannot control the influence of the domain shift problems on other modules, we simulated a situation that only the slot filling module was influenced by the domain shifts when encountering unseen domains. Additionally, we maintain other modules to be able to deal with unseen domains.

To do so, we set the dialogue system as follows: Only for the slot filling module, we trained the models by considering seen and unseen domains, as described in Section 3.1. We did not change other modules so that they could still deal with all possible domains based on pretraining or pre-defined rules and templates for specific datasets.

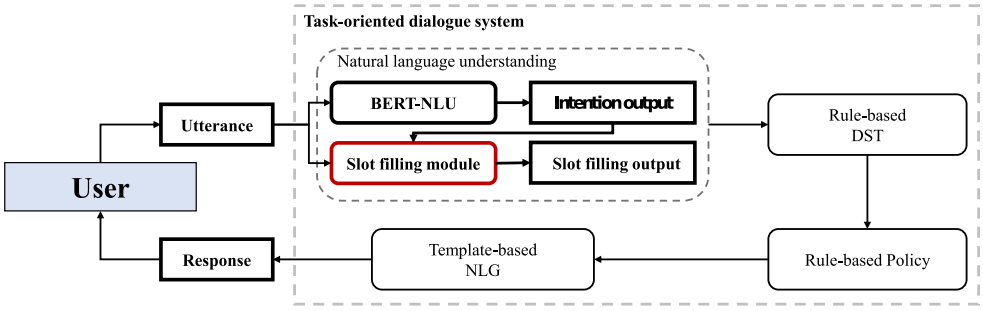

Figure 6.1: The process of the constructed task-oriented dialogue system

The process of our system is shown in Figure 6.1. In the process, we first obtained the BERT-NLU's results containing the domain, intention, slot, and value. The domain and intention here indicate the task domain and user's intended actions, such as 'Attraction-Inform' and 'Hotel-Request.' Generally, zero-shot slot filling approaches play the role of extracting the values from given utterances, which corresponds to the action 'Inform.' Accordingly, we performed the following process based on three situations: (i) For the 'Request' action, the slot filling module is not required to identify the values for slots from given utterances. We maintained the full results of (domain, intention, slot, value) that contained the 'Request' actions from the BERT-NLU; (ii) Some slots belonging to the 'Inform' action were not required to be identified from given utterances, such as the slot 'internet' in the Hotel domain, which requires the NLU module to determine whether there is internet service. We also maintained the results containing such slots from the BERT-NLU; (iii) Except for these two situations, we discard the slot and value results and maintain the domain and intention results from BERT-NLU. Then, we use the replaced slot filling module to obtain the slot filling results for each maintained domain and combine the domain-intention results with the slot filling results as the output results.

Afterward, we fed the results from all three situations to subsequential modules. Finally, the system generated and provided responses to the user.

6.3 Experiment

6.3.1 Dataset

To evaluate the dialogue systems based on the settings described above, as well as to guarantee that the dialogue situation is close to the real world, we conducted our experiments on the MultiWOZ dataset [88]. In

the dataset, the user and system generally complete dialogue with multiple turns across multiple domains. Meanwhile, the domains in the dataset are often encountered in daily life. In addition, the ConvLab-2 platform has the pre-trained BERT-NLU and pre-defined rule-based DST, rule-based policy, and template-based NLG modules that are adapted to the MultiWOZ dataset. The performance of the system using these modules was shown to be much better than that using other modules [14] on the MultiWOZ dataset. Therefore, conducting our experiments on the MultiWOZ could be benefited from these modules' accuracy to precisely investigate the influence of using zero-shot slot filling for the dialogue system when encountering unseen domains.

6.3.2 Zero-shot slot filling training

To simulate the situation of encountering unseen domains, we trained zero-shot slot filling and slot filling module on the MultiWOZ 2.3 dataset, which is the latest version of the MultiWOZ dataset. We followed the same zero-shot settings in our previous experiments to train all zero-shot slot filling models but with a little modification.

For the zero-shot methods, in previous experiments, we used a part of the zero-shot domain's data as the validation set to choose the best model and used the remaining as the test set. To simulate the zero-shot scenario in the real world, where the zero-shot domain's data is unavailable, we did not split the zero-shot domain into the validation set and the test set. Specifically, we used the last 20% instances from each training domain as the validation set for choosing the best model. We used the zero-shot domain's data as the test set alone to evaluate the performance of zero-shot slot filling models. Then, each model was used to construct a complete dialogue system following the process described in section 6.2.

6.3.3 Dialogue system experiment setting

To evaluate the dialogue system's performance, we employed the evaluation process from ConvLab-2. In the evaluation process, another pipeline dialogue system was used as the user agent to act as a user. The user agent was based on the same pipeline system as the base system, which also consisted of BERT-NLU + rule-based dialogue state tracking + rule-based policy + template-based natural language generation.

In the experiments, the dialogues between the user agent and the dialogue system were conducted as follows:

1. At the beginning, the user agent obtained the user's goals from the pre-defined constraints in the MultiWOZ dataset.
2. Then, the user agent generated utterances based on the goals to ask the system.
3. The system received and processed the given utterances and generated responses.
4. The user agent received the system's utterances and processed them.
5. The user agent generated new goals based on the dialogue and generated the dialogues for the system.

Then the dialogues were repeated from step 3 to step 5. When the user agent determined that the dialogue was completed or the dialogue reached the pre-defined max turn, the dialogue was over. In this study, we used the default setting for the max turn 40. For every dialogue, an analyzer recorded the user's goals and dialogues for evaluation.

Since we trained six separate models by setting each domain in the MultiWOZ 2.3 dataset as the zero-shot domain and trained three times for each separate model, we constructed 18 dialogue systems for one slot filling method. We evaluated them separately to obtain the performances. For each system, we conducted 100 dialogues. Each dialogue consisted of multiple turns. We conducted the experiments for each system three times to reduce the influence of random initialization. Consequently, we conduct 5400 dialogues for each slot filling method in our experiment. The average performance of the dialogue system was used for evaluation.

6.3.4 Evaluation metrics

To evaluate the performance of zero-shot slot filling, we applied the widely used metric slot F1 score.

In evaluating the dialogue system performance, we applied a metric to evaluate the performance of slot filling in the dialogue process [14]:

Slot F1 in dialogue: To investigate the relationship between the performance of zero-shot slot filling and the performance of slot filling through the dialogue process in which unseen domain is encountered, we used slot F1 in dialogue to evaluate the slot filling during the dialogue process. We compute the slot F1 in dialogue for the NLU results of case (3) described in section 6.2. We did not count the results of case (1) and case (2) since we did not perform slot filling by the replaced slot filling modules for these two cases.

Complete rate: The ratio of the dialogues in which user goals were completed by the system. The user goals generally indicate the slots that

need to be requested. For instance, in the Hotel domain, a user may request the information on the hotel's price or area. Those goals are generated by the user simulator. If the system answered all user's requests after a dialogue and filled the unknown values in the user's request states, the dialogue is treated as a completed dialogue. On the other hand, if the user's request states still contain unknown or uncertain values when the dialogue ends, the dialogue is not completed. For instance, the initial user's goal for the Hotel domain is (Request, price, ?) and (Request, phone, ?). When the user asked the system about the price, and the phone about a hotel and the system answered with correct information, the user agency will record those answers to change the uncertain value "?" into specific values, such as "area-west" and "phone-0123456". In this case, the dialogue is completed. The completion rate is computed by dividing the total number of dialogues by the number of completed dialogues.

Success rate: The ratio of the dialogues in which all user requests were informed, and the booked entities satisfied the pre-defined constraints in the dataset. Accordingly, judging whether a dialogue is 'success' or not contains three aspects. The first one is that the dialogue system informed all requested components by users. This aspect is similar to the judgment of dialogue completion but more focused on the dialogue contents. The dialogue completion considers whether the user's goals were clear, and the success focuses on whether the requested components within the dialogue were informed. The second and third ones are whether the booked entities satisfy the constraints and whether the domains of the booked entities satisfy the constraints. Since we simulate a dialogue process by employing user agency, the constraints could be obtained for each dialogue from the definitions in the dataset. The booked entities indicate what entities the dialogue system booked for the user; the domains of the booked determine which domain the system performed booking. These two aspects reflect whether the system correctly performed booking for the user. For a dialogue, if the system informed all requested components correctly and booked that satisfied the pre-defined constraints of entities and domains, the dialogue is treated as 'success.' Otherwise, the dialogue is not successful. The success rate is computed by dividing the total number of dialogues by the number of the success dialogues.

Inform F1: The metric in evaluating how the dialogue system correctly informed information. In the computation process of inform F1, if we treat the requests in the user's goal as the golden truths, the requests that were informed correctly by the system are counted as true positive samples (TP). While the requests that were not informed are counted as false negative samples (FN). The system incorrectly informed requests, which are the

requests that the user did not need to request, are the false positive samples (FP). Then the inform F1 can be computed as follows:

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6.1}$$

Book rate: The ratio of booked entities satisfied the pre-defined user constraints. This metric is specific to evaluate how the bookings were performed. Although the success rate considered the correctness of booked entities, since the booking actions are not necessary for all dialogue scenarios, the booking rate is used to exactly evaluate the booking accuracy. The booking rate is computed by dividing the number of all necessary booking contents by the number of satisfied booked entities.

For all metrics, one-way ANOVA is used to examine whether the difference is significant or not between every two methods or systems.

6.3.5 Comparison methods

For our purpose, we used three kinds of methods as the slot filling module in the dialogue system.

Zero-shot slot filling methods: We employed our best model of combining the multi-relation-based and ontology-based representations as a method for replacing the slot filling module. For convenience, we name our model the intrinsic representation approach (IRA). We also used CT [19] and Coach [22] as the representative methods from the one-stage approach and the two-stage approach, respectively. For the dialogue system using these zero-shot methods, we followed the process described in Section 6.2.

Conventional methods: To investigate whether zero-shot methods contribute to the dialogue system compared with conventional methods, we used BiLSTM-CRF [9] method to construct baseline systems. Generally, unlike zero-shot methods that could predict slots for each domain maintained from NLU's results, conventional methods could not perform slot filling for specific domains. To employ the BiLSTM-CRF method for comparison, we designed two training and predicting processes:

1. We used the output label form (domain-IOB-slot) to train the model. Thus, the model can simultaneously predict slots and domain names. In slot filling evaluation on zero-shot target domains, since the ground truth labels do not include domain names, we use the model to predict the labels formed as (domain-IOB-slot) and discard the 'domain' component from the results for evaluation. In the process of the dialogue system, for the three cases described in Section 2, we did not change

the process for case (i) and case (ii). For case (iii), we further discarded the domain names in BERT-NLU results. Then, we used the BiLSTM-CRF model to predict domain, slot, and value results and combined the action 'Inform' as the NLU's output in the dialogue system.

2. Since process (1) can only predict the domain names that appear in training and cannot predict unseen domain names, the model with process (1) will not correctly predict any results for unseen domains. This may make the comparison ambiguous since we could not know whether the difference in system performances was due to the domain name prediction or slot filling. We designed another training and predicting process to make the comparison clearer for our purpose. Specifically, we used the label form (IOB-slot) to train the model, where the model was not required to predict domain names. Although the model still cannot predict unseen slots, the model can predict seen slots in unseen domains.

   In slot filling evaluation, we directly used the labels formed (IOB-slot) for evaluations. In the process of the dialogue system, for case (iii) in Section 2, we first obtained all possible domains that appeared in the utterance, which is similar to our action for zero-shot methods. Then, we used the BiLSTM-CRF model to obtain slot filling results. Next, we attempted to match each slot in the results to possible domains to find whether the slot belonged to any domain. If only one domain was matched, we combined the slot-value result with the domain-action (domain-Inform) result as the output; If multiple domains were matched, we randomly selected one domain as the matched domain and combined the slot filling result with the domain-action result as the output; If no domains were matched, we discard the slot filling results.

According to the process settings, process (1) could be expected to maintain close performances of original BiLSTM-CRF on handling seen domains but could not deal with unseen domains. On the other hand, process (2) improved the robustness of handling unseen domains if the unseen domains contained seen slots. Moreover, process (2) could be expected to improve the robustness for seen domains because the slot set space was smaller in process (2) than in process (1). The reason is that when seen domains have overlapped seen slots, the overlapped slots in different domains are treated as the same class in process (2) but are treated as different classes in process (1).

The ideal method: Finally, we investigated whether the constructed dialogue systems are generally well-performed or not compared to an ideal

system with a perfect slot filling module. To realize the ideal system, in the simulation process described above, we could access the ground-truth NLU results generated by the user's agency. We use the ground-truth results as the NLU results in the ideal dialogue system. Accordingly, the ideal system will have a perfect prediction for slot filling and domain detections. The ideal system would only be investigated for comparison with the dialogue systems' performances and will not be compared in zero-shot slot filling performance.

In summary, we compared six dialogue systems using different slot filling modules or slot filling settings, including two conventional methods based on BiLSTM-CRF, three zero-shot methods of CT, Coach, and IRA, and an ideal system with a perfect slot filling module.

6.4 Results and discussions

6.4.1 Experimental results and discussions

We first investigated the zero-shot slot filling performance. Figure 6.2 shows the domain average performance of zero-shot slot filling of different methods. P1 and P2 indicate the BiLSTM-CRF model using process (1) and process (2) in Section 3.4. According to the one-way ANOVA, the differences between every two methods, except between P1 and P2, were significant on the level of $p < 0.001$. While the difference between P1 and P2 was not significant. The average results of P1 and P2 show that P2 outperformed P1 by 0.55 on the average slot F1 score. The results of zero-shot methods and better conventional model P2 show that Coach achieved an improvement of 9.94 on the average slot F1 score. CT and IRA further improved P2 by 14.46 and 18.70 on average slot F1 scores, respectively. These results confirmed that by alleviating the domain shift problems, zero-shot methods more effectively handle unseen domains than conventional methods.

Next, we investigated whether zero-shot slot filling benefited the dialogue system. Figure 6.3 shows slot F1 in dialogue performances of different methods. In Figure 6.3, according to one-way ANOVA, the difference between every two methods was significant on the level of $p < 0.001$ except for the difference between the P2 and Coach, which was not shown a significant difference. For clear looking, we did not make annotations of significant levels. Figure 6.4 shows four dialogue performance metrics, including complete rate (a), success rate (b), inform F1 (c), and book rate (d). In Figure 6.4, the '*,' '**,' '***' annotated between the two methods showed the significant difference levels between the two methods according to one-way ANOVA, which are $p < 0.05$, $p < 0.01$, and $p < 0.001$, respectively.

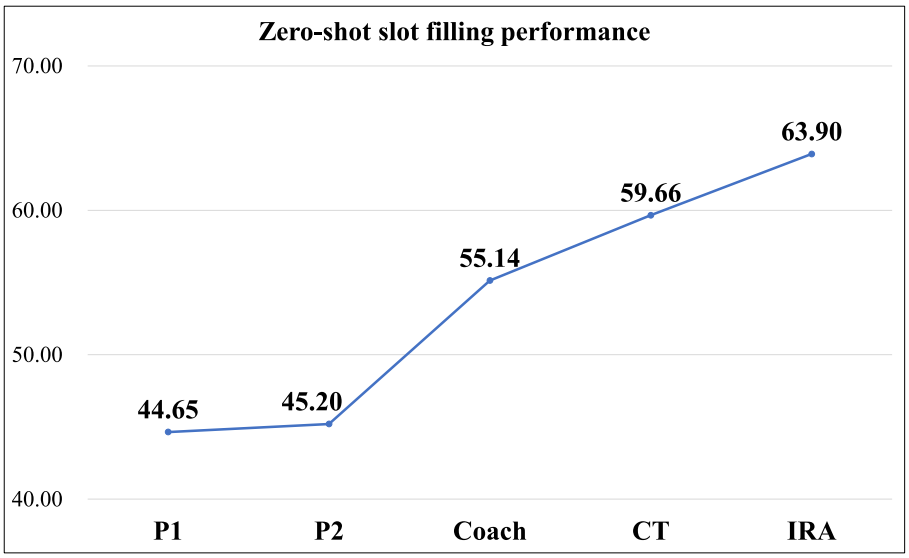

Figure 6.2: Average performance in zero-shot slot filling of different methods

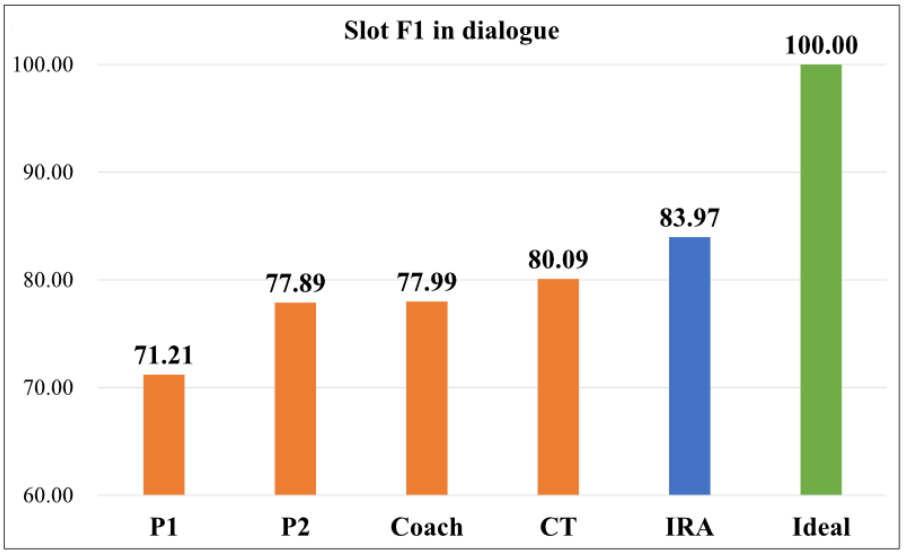

Figure 6.3: Average performance on slot F1 in dialogue by dialogue systems using different slot filling modules

By comparing the results in Figure 6.2 and Figure 6.3, with the improvement of 9.94 on zero-shot slot filling, the system using Coach improved the system using P2 by 0.10 on slot F1 in dialogue. Next, with the improvement of 14.46 on zero-shot slot filling, the system using CT improved P2 by 2.2 on slot F1 in dialogue. Finally, with the improvement of 18.70 on zero-

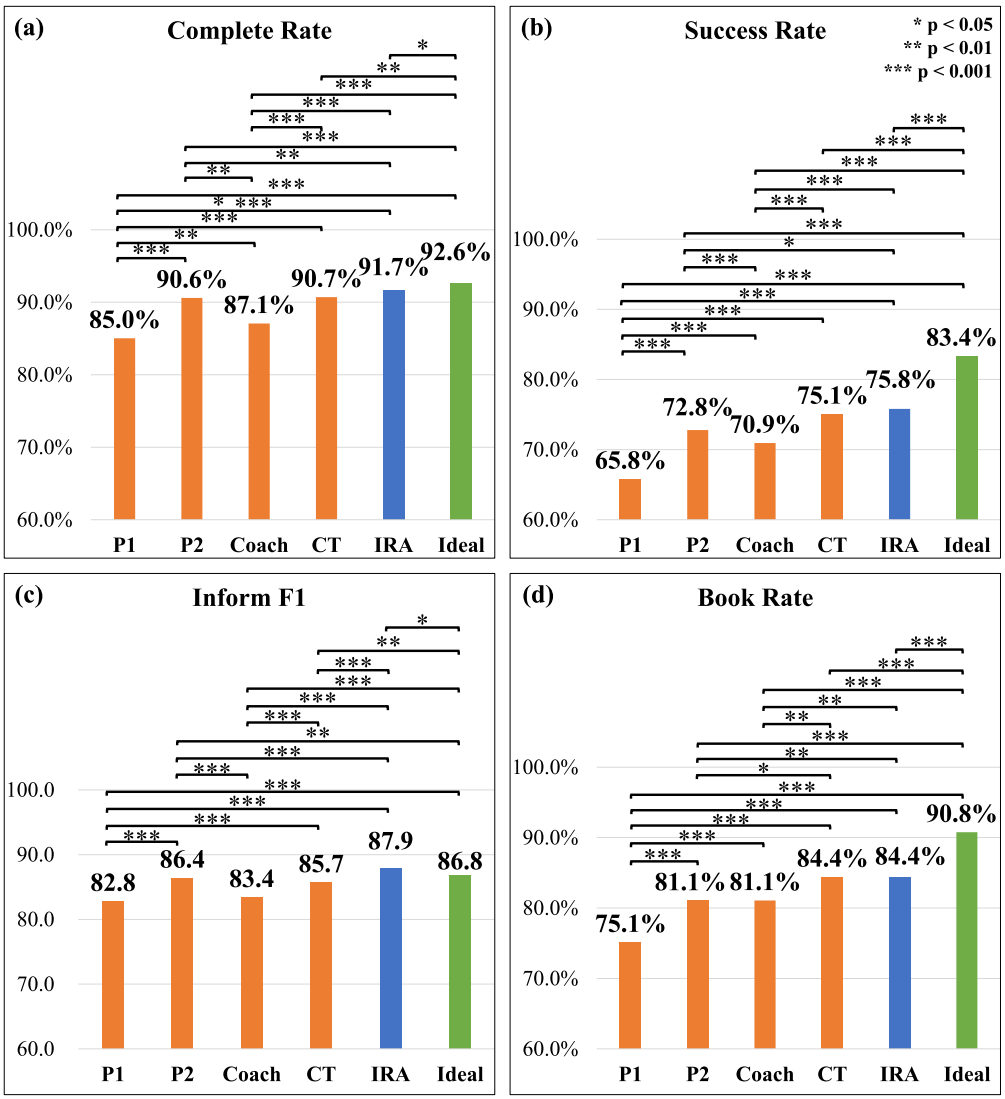

Figure 6.4: Average performance on dialogue evaluation metrics by dialogue systems using different slot filling modules. (a) Performance on the complete rate; (b) performance on the success rate; (c) performance on the inform F1; (d) performance on the book rate

shot slot filling, IRA further improved P1 by 6.08 on slot F1 in dialogue. These results demonstrated that the improvement of zero-shot slot filling generally contributes to the performance of slot filling in dialogue systems when encountering unseen domains. Note that the zero-shot methods improved P2 relatively less on slot F1 in dialogue than on zero-shot slot filling. The reason is that the system faced utterances about unseen domains

as well as utterances about seen domains through dialogues. Therefore, the advantages of zero-shot methods on zero-shot target domains were not completely reflected in slot filling through dialogues.

The performances in Figure 6.3 and Figure 6.4 show that although the ideal system performed perfectly on slot filling with a slot F1 in the dialogue of 100, the ideal system did not outperform all others on inform F1. Moreover, Coach was defeated by P2 on complete rate and inform F1 by 3.5% and 3%, respectively, although Coach and P2 had close performances on slot F1 in dialogue. CT was defeated by P2 on inform F1 by 0.7%, although CT outperformed P2 on slot F1 in dialogue. Accordingly, complete rate and inform F1 performances showed different tendencies to slot F1 in dialogue.

To find the reasons for these results, we fixed one random seed to make all dialogue systems face the same user goals and analyzed the dialogue logs. We found that in some dialogues, the user agent did not ask about all the contents that should be requested. For instance, in a dialogue about booking a restaurant and hotel, the pre-defined goals contained the requests for the phone number and address of a restaurant and the phone number of a hotel. These goals should be requested in the dialogue so that the dialogue system can give the user information to help the user to complete the booking. However, the user agent asked about the phone number of the hotel in some cases but did not in other cases. One reason could be considered that the dialogue process is different due to the dialogue state tracking and dialogue policy in both the user agent and the dialogue system sides. For instance, with the same goal setting mentioned above: in one case, the user agent asked about the hotel in the north area, but the dialogue system did not find any satisfied candidates, so the user agent did not further ask about the hotel's phone number; in another case, the user agent asked about the hotel in the north area but had a different context. The dialogue system found candidates for the user, so the user further asked for the phone number, and the system responded to that. In the latter case, the dialogue was determined as 'complete' because the system informed all contents that should be requested. The former case was not 'complete' because the hotel's phone number was not informed. Similar to the complete rate, for the inform F1, if the user agent did not ask about all requests in the pre-defined goals, the system could not inform about those requests, which degraded the inform F1. These results explain the inconsistency between the complete rate and slot F1 in dialogue and between inform F1 and slot F1 in dialogue. These results also suggested that the complete rate and inform F1 are sensitive to other modules rather than slot filling. The errors of other modules can degrade the entire system, even if the slot filling module does a good job. Therefore, although improving the slot filling module contributes to the

system performance, such an improvement may not consistently improve all aspects due to the limitations of other modules.

On the other hand, the success rate and book rate showed more consistent tendencies with slot F1 in dialogue than the complete rate and inform F1. The only inconsistent result was between Coach and P2. As shown in Figure 6.3, Coach and P2 had close performances on slot F1 in dialogue, while in Figure 6.4 (c), Coach was slightly defeated by P2 on success rate. By analyzing and comparing the dialogue logs between Coach and P2, we found the main reason to be that the Coach's errors usually led the dialogue system to mismatch values and slots. For instance, for the utterance 'I also need a hotel. It should have free WIFI. It doesn't need to have parking. I would like a 3 star hotel though, please.' Coach incorrectly predicted the number '3' as the hotel type, while the correct slot of '3' should be the number of stars of the hotel. The error caused by the Coach made the system provide incorrect information or unable to find satisfactory candidates for the user. Thus, this type of error may break the dialogue before the user makes all requests. On the other hand, P2's errors were usually missing predictions. For instance, for the same utterance above, the dialogue system using P2 correctly predicted the 'hotel' as the hotel type but did not make any prediction for '3'. The error caused by P2 can make the system provide candidates with uncompleted constraints, but the user still has the chance to ask about the points that were not satisfied (although the user may not ask), so the dialogue system can complete the dialogue and satisfy the user's goals. Therefore, Coach was defeated by P2 on the success rate. Moreover, CT and IRA had a few errors of mismatching values and slots. Other errors by CT and IRA were similar to P2. Therefore, the performance of CT and IRA showed consistent tendencies in the success rate and book rate to slot F1 in dialogue. These results suggested that alleviating the error of mismatching slots and values in zero-shot slot filling can be important in improving the performance of the dialogue system.

Finally, towards the ultimate goal, the domain adaptable task-oriented dialogue system need to be as good performance on unseen domains as the system that trained on unseen domains. We compared the performance of the system using IRA module with the base system (BERTNLU + rule-based DST + rule-based DP + template-based NLG) to investigate the performance difference between a domain adaptable system with a well-performed conventional system to show the improvement directions in future. The slot filling performance of BERTNLU was reported as 89.03 on the test set containing all domains of MultiWOZ 2.3 dataset [88]. Although this performance was obtained from a different experiment setting to our research, BERTNLU can be considered generally better than IRA on unseen domains since BERTNLU

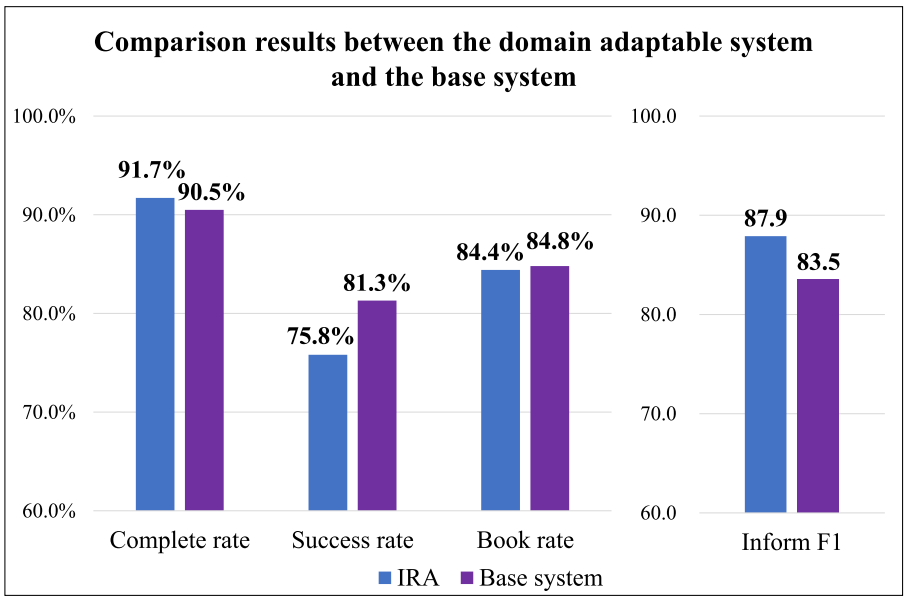

Figure 6.5: Comparison results between the domain adaptable system using IRA and the base system

was trained on all domains. Figure 6.5 shows the comparison result between systems. As seen in the figure, the system using IRA outperformed the base system on complete rate and inform F1. However, as discussed above, these two metrics were sensitive to other modules besides the NLU, so the base system may suffered from the limitations of other modules. The better performance of the system using IRA cannot demonstrate that the domain adaptable system was better than conventional system. Nonetheless, these results suggested that the bottleneck for the system using IRA is on other modules rather than slot filling. It is important to comprehensively improve the system by considering all modules and their combined effects. On the other hand, the domain adaptable system using IRA was defeated by the base system on success rate and book rate. This is because these two metrics are related to the NLU module's performance, and IRA was not as good performance as BERTNLU on unseen domains. These results suggested that developing domain adaptable system by improving the zero-shot capacity for unseen domains is still necessary towards the ultimate goal of the task-oriented dialogue system.

Unseen domain: Attraction

Dialogue system using conventional methods

<Dialogue begin>
<user in []>
<user out ['Inform', 'Attraction', 'none', 'none'], ['Inform', 'Attraction', 'Type', 'swimmingpool'], ['Inform', 'Attraction', 'Area', 'centre']>
<sys in ['Inform', 'Attraction', 'Area', 'centre']>
<sys out ['Inform', 'Attraction', 'Choice', '44'], ['Recommend', 'Attraction', 'Name', 'club salsa']>
User: I am looking for tourist attractions. How about a swimmingpool? I ' m looking for some attractions in the centre.
Sys: Sure, there are 44 in that area. Club salsa meets your requirements.

<user in ['Inform', 'Attraction', 'Choice', '44'], ['Inform', 'Attraction', 'Name', 'Club salsa']>
<user out ['Request', 'Attraction', 'Address', '?']>
<sys in ['Request', 'Attraction', 'Address', '?']>
<sys out ['Inform', 'Attraction', 'Address', 'park street']>
User: That sounds great! May I have the address of the attraction?
Sys: The address is park street. ←The address of unsatisfied attraction
…

<Dialogue result: Success ×>

Dialogue system using zero-shot methods

<Dialogue begin>
<user in []>
<user out ['Inform', 'Attraction', 'none', 'none'], ['Inform', 'Attraction', 'Type', 'swimmingpool'], ['Inform', 'Attraction', 'Area', 'centre']>
<sys in ['Inform', 'Attraction', 'Type', 'swimmingpool'], ['Inform', 'Attraction', 'Area', 'centre']>
<sys out ['Inform', 'Attraction', 'Name', 'Parkside pools']>
User: I am looking for tourist attractions . How about a swimmingpool? I ' m looking for some attractions in the centre.
Sys: Parkside pools is popular among visitors.

<user in ['Inform', 'Attraction', 'Name', 'Parkside pools']>
<user out ['Request', 'Attraction', 'Address', '?']>
<sys in ['Request', 'Attraction', 'Address', '?']>
<sys out ['Inform', 'Attraction', 'Address', 'Gonville place']>
user: Can I get the address of the attraction?
sys: Their address in our system is listed as Gonville place. ←The address of satisfied attraction
…

<Dialogue result: Success √>

Figure 6.6: Example of explaining how zero-shot methods benefit to the dialogue system

6.4.2 Case study

In this section, we give an example in Figure 6.6 to show the most benefit of zero-shot methods to the dialogue performance. In Figure 6.6, the 'Attraction' domain is the unseen domain. Additionally, 'user in' and 'sys in' indicate the NLU outputs of the user agent and dialogue system, respectively, and 'user out' and 'sys out' are the goals generated by the user agent and dialogue system to generate the next utterance. 'User' and 'Sys' are the dialogue logs. As seen in the figure, after the user requested an attraction, the user informed detailed needs of the attraction. The user's detailed information is usually related to the detailed requests, which is important for making a satisfactory booking and completing all requests of the user. The dialogue system using conventional methods missed the prediction for the slot 'Type' for detailed needs, so the dialogue system recommended an attraction

that did not satisfy all requests of the user, which made the dialogue system finally fail to satisfy the user's requests and made the dialogue unsuccessful. On the other hand, the system using zero-shot methods correctly understood the details based on the correct slot filling results, so the system gave a satisfactory recommendation and finally filled all requests of the user to make the dialogue successful. This case study demonstrated that by alleviating domain shift problems for handling unseen domains, the largest benefit of using zero-shot methods in the dialogue system is to avoid miss predictions to satisfy all requests of the user.

6.5 Summary

In this chapter, we investigated whether zero-shot slot filling benefits the task-oriented dialogue system when encountering new domains that contain unseen slots. Furthermore, we investigated whether the improvements in zero-shot slot filling contribute to the performance of the dialogue system. The results demonstrated that by handling slots well in unseen domains, zero-shot methods generally improved the slot filling in dialogues and consistently improved the success rate and book rate. Nonetheless, due to the limitations of other modules in the dialogue process, the improvements by zero-shot slot filling sometimes cannot consistently improve the performance of the dialogue system in specific aspects, such as the complete rate and inform F1. The results also suggested that alleviating the errors of mismatching values and slots in slot filling can improve the performance of dialogue systems. These results also confirmed that our research on mining intrinsic representations to improve zero-shot slot filling is an effective approach towards the ultimate goal of the task-oriented dialogue system.

Chapter 7

Conclusion and future works

7.1 Summarization of the research

This research aimed to address domain shift problems towards the ultimate goal of the task-oriented dialogue system, which is to build a domain-adaptable system that can perform zero-shot adaptation to any given domain without training instances. Specifically, we focused on the zero-shot capacity of the slot filling module in the dialogue system since the intended components extracted by the slot filling module are the basis of the subsequential processes in the dialogue system. For our purpose, we mined the intrinsic representations that describe the intrinsic characteristics of slots, values, and the relationships between slots and values for dealing with the domain shift problems in zero-shot slot filling. These intrinsic representations were expected to provide transferable information across domains effectively.

In this research, we first proposed inference relation paths (IRPs) from the knowledge graph on dealing with the domain shift problem of semantically dissimilar unseen slots. We found that IRPs implicitly carry the relationships between slots and their values by analyzing a number of IRPs. We used IRPs as slot descriptions with the one-stage approach for zero-shot slot filling. Experimental results demonstrated that IRP slot descriptions provided effective transferable information compared to using conventional slot descriptions alone. Further analysis showed that IRP slot descriptions were more effective than conventional descriptions in dealing with unseen slots. However, although IRP alleviated domain shift problems of unseen and seen slots, the results showed that the absolute improvements were not high. Moreover, obtaining IRPs was not flexible since the knowledge graph usually does not cover all possible slots and values in practice. In addition, it is hard to analyze and explore intrinsic representations under the interpretability limitation of the one-stage approach.

To overcome the limitations of IRPs and the interpretability limitation of the one-stage approach, we proposed the multi-relation-based representation to alleviate the domain shift problems to deal with the domain shift problem

of different context distribution with the two-stage approach. Specifically, we focused on the slot entity prediction. The slot entity prediction is the premise of overall zero-shot slot filling, so improving slot entity predictions is expected to improve zero-shot slot filling. The multi-relation-based representation captures the entity-slot relation by considering all possible slots in a given domain and the entity-context relation by considering various context environments. Therefore, the proposed representation was expected to describe intrinsic characteristics of slot entities to alleviate the domain shift problem. Moreover, the proposed representation could be obtained from the given slots and utterances and thus was more flexible than IRPs. Then, we constructed a model to utilize the multi-relation-based representations in zero-shot slot filling. Experimental results demonstrated that the proposed representation was effective on predicting slot entities from the context corresponding to unseen slots as well as the context corresponding to differently explained seen slots. Further analysis showed that by alleviating domain shift problems in slot entity predictions, the proposed representation improved zero-shot slot filling compared to previous studies. However, the improvements on unseen slots were not high, and the limitations of slot type predictions cannot bring out the advantages by the improvements of slot entity predictions.

To alleviate domain shift problems on unseen slots and seen slots in slot type predictions, we proposed ontology-based representations to fill the knowledge gap between domains. The ontology is a knowledge base that describes the relationship between slots and values. With the assumption that the ontology is fully-defined for both unseen and seen domains, the ontology would not change when the domain shifts from one to another, thus describing intrinsic relationships between slots and values across domains. We proposed two methods to use the ontology for zero-shot slot filling: using the ontology as a text-matching base and using the ontology for constraints. The results demonstrated that the two methods both improved zero-shot slot filling compared to previous studies, and using the ontology for constraints was more robust at distinguishing the slots that have shared values. We further combined the multi-relation-based representation and the ontology-based representation to alleviate domain shift problems. The results demonstrated that the combination of two representations further alleviated domain shift problems and significantly improved zero-shot slot filling. Further analysis showed that the proposed representations significantly improved the predictions on unseen slots as well as differently explained seen slots.

With the three proposed intrinsic representations, we alleviated domain shift problems on unseen slots, differently explained seen slots, and different context distributions. We summarize the domain average results in Table 7.1. In the table, I indicates the performance using IRPs, M indicates the

Table 7.1: Summarization of the performance of using intrinsic representations

Dataset	I	M	O	M+O
Snips2017	44.33	43.35	-	72.83
Snips	-	39.16	61.64	68.12
MultiWOZ 2.3	-	62.32	67.67	69.10
SGD	-	69.59	-	79.18

performance using the multi-relation-based representation, O indicates the performance using the ontology-based representation, and M + O indicates the performance of the combined use of the multi-relation-based representation and the ontology-based representation.

As seen in Table 7.1, the performances by using IRPs and using the multi-relation-based representation were on a similar level, while using the ontology-based representation significantly improved the performance. The combined use of the multi-relation-based and ontology-based representations further improved zero-shot slot filling. From these results, we can summarize that the ontology-based representation is the most effective one since it establishes the relationships between slots and values, which is important for handling unseen slots as well as seen slots. The limitation of using the ontology is that it needs to be fully defined in our research. How to deal with the situation that the ontology is not fully defined for all domains is considered to be a future work. Compared to the ontology-based representation, the multi-relation-based representation's strength is to predict slot entities correctly. With the improvement in slot entity predictions, the advantages of the ontology were more brought out. The limitation of the multi-relation-based representation is that when the slot sets increase, the computation complexity will increase. How to use a limited set of slots to compute the multi-relation-based representation could be a future work to reduce computation complexity. Compared to the two representations above, the IRP took commonsense knowledge into account, which makes IRPs potentially handle unseen slots in a similar way as humans do. However, it is not flexible to extract IRPs based on present technologies. Developing the representations of knowledge graphs that can be used to extract IRPs between arbitrary entities could be expected to alleviate the inflexibility of IRPs.

On the other hand, the results show that the best performances using the intrinsic representations still have a gap to the perfect slot filling. To further alleviate domain shift problems to improve zero-shot slot filling, it is necessary to clarify how domain shift problems influence the model

performance more quantitatively. However, it is hard to analyze the role of each component of domain shift problems in zero-shot slot filling so far. For instance, by revisiting the performance of each specific slot in three datasets, we found that the performance on the unseen slot 'playlist owner' from the Add To Playlist domain was low, which was about 0.38 on the slot F1 score. The reason for the low performance can be that the slot is unseen, but the performances on other unseen slots were not as low as the 'playlist owner.' The reason for the low performance could also because the context situation of 'playlist owner,' but the context situations of other unseen slots were also diverse for the model. It is hard to analyze how domain shift problems influenced the performance of the slot 'playlist owner.' Therefore, developing the methods of quantitatively analyzing domain shift problems can be effective for further analyze domain shift problems to improve zero-shot slot filling.

After improving zero-shot slot filling by intrinsic representations, whether the improvements on zero-shot slot filling contribute to the performance of the whole dialogue system was still not clear. For this problem, we investigated the performances of different dialogue systems encountering zero-shot domains. We constructed dialogue systems using different slot filling modules, including conventional method-based modules and zero-shot method-based modules. The investigation results demonstrated that the dialogue systems using zero-shot slot filling modules were more effective than those using conventional slot filling modules when encountering unseen domains. Furthermore, the improvements in zero-shot slot filling generally contributed to the performance of dialogue systems from different aspects, although some specific aspects did not consistently improve with the improvements of zero-shot slot filling. Therefore, improving zero-shot slot filling could be an approach towards the ultimate goal of the task-oriented dialogue system.

7.2 Conclusion

In conclusion, towards the ultimate goal of the task-oriented dialogue system, this research mined three intrinsic representations from multiple aspects to address domain shift problems of unseen slots, differently explained seen slots, and different context distributions. Specifically, this research mined the inference relation path representation to describe implicit relationships between slots and values, the multi-relation-based representation that captures the general and specific characteristics of slot entities, and the ontology-based representation that describes intrinsic relationships between slots and

values. Experiments demonstrated that the proposed representations effectively alleviated domain shift problems on zero-shot slot filling. This research further confirmed that the improvements in zero-shot slot filling contributed to improve the performance of the whole dialogue system. Therefore, by alleviating the domain shift problems in zero-shot slot filling, the proposed representations were effective towards the ultimate goal of the task-oriented dialogue system.

7.3 Future work

Towards the ultimate goal of the dialogue system, this research concentrated on alleviating domain shift problems by mining intrinsic representations. Future studies may be improved from the following aspects:

1) Methods for analyzing domain shift problems

As discussed in Section 7.1, it is necessary to analyze the influence of domain shift problems on specific slots to clarify the reasons for low performances. However, it is hard to quantitatively clarify the influences so far. Therefore, developing the methods for clarifying the domain shift problems on each specific slot can be expected to be a tool to further alleviate domain shift problems for improving zero-shot slot filling. One idea is to quantitatively statistic the difference of vocabularies and utterance patterns between source domains and target domains to clarify the influence of the difference in context distributions. Another idea is to compare the mentioning patterns of unseen slots and training slots to clarify the difference between unseen slots and seen slots besides semantical similarity.

2) Obtaining knowledge from pre-trained models

Since a real understanding of languages is considered necessary to handle unseen contents as people do, the knowledge, such as commonsense knowledge defined in the knowledge graph, is necessary for zero-shot slot filling. However, as discussed in Section 7.1, the knowledge from the knowledge base, such as the inference relation path used in this research, is not flexible to be used in practice. Therefore, finding a flexible knowledge representation to describe the relations between slots and values is necessary. One idea is to fine-tune a knowledge-enhanced pre-trained model with the task of optimizing the representations of entity relationships.

3) Methods dealing with non-fully-defined ontology

The ontology was useful to constrain the slot candidates in zero-shot slot filling. However, although the ontology is assumed to be fully-defined to include all possible values for a slot, new values can be appearing fast in practice. Therefore, developing a method to deal with incomplete ontology

could be better than relying on a complete ontology.

4) Multimodal methods for zero-shot dialogue system

Human interactions in dialogue not only rely on linguistic information, but also rely on paralinguistic and nonlinguistic information. Other modalities besides text could also be effective in providing transferrable information across domains to complement the limitations of using text alone. This can be expected to further alleviate domain shift problems when adapting the dialogue systems to new domains. Therefore, taking audio, vision, and other modalities into account and developing a multi-modality dialogue system could be an approach towards the ultimate goal of the task-oriented dialogue system.

References

[1] Weizenbaum, J. (1966). ELIZA-a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1), 36–45.

[2] Wallace R S. The anatomy of ALICE[M]//Parsing the turing test. Springer, Dordrecht, 2009: 181-210.

[3] Bobrow, G., Kaplan, R. M., Kay, M., and Winograd, T. (1977). GUS , A Frame-Driven Dialogue System, 155–173.

[4] Zue V, Glass J, Goodine D, et al. The VOYAGER speech understanding system: Preliminary development and evaluation[C]//International Conference on Acoustics, Speech, and Signal Processing. IEEE, 1990: 73-76.

[5] Wu C H, Yan G L, Lin C L. Spoken dialogue system using corpus-based hidden Markov model[C]//Fifth International Conference on Spoken Language Processing. 1998.

[6] Mikolov T, Karafiát M, Burget L, et al. Recurrent neural network based language model[C]//Interspeech. 2010, 2(3): 1045-1048.

[7] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks[J]. Advances in neural information processing systems, 2014, 27.

[8] Kenton J D M W C, Toutanova L K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[C]//Proceedings of NAACL-HLT. 2019: 4171-4186.

[9] Budzianowski P, Vulić I. Hello, It's GPT-2-How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems[C]//Proceedings of the 3rd Workshop on Neural Generation and Translation. 2019: 15-22.

[10] Zhang Z, Takanobu R, Zhu Q, et al. 2020. Recent advances and challenges in task-oriented dialog systems [J]. Science China Technological Sciences.

[11] Pei J, Ren P, Monz C, et al. Retrospective and Prospective Mixture-of-Generators for Task-Oriented Dialogue Response Generation[M]//ECAI 2020. IOS Press, 2020: 2148-2155.

[12] Campagna G, Foryciarz A, Moradshahi M, et al. Zero-Shot Transfer Learning with Synthesized Data for Multi-Domain Dialogue State Tracking[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020.

[13] Yang Y, Li Y, Quan X. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2021, 35(16): 14230-14238.

[14] Zhu Q, Zhang Z, Fang Y, et al. 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations.

[15] Kouw W M, Loog M. An introduction to domain adaptation and transfer learning[J]. arXiv preprint arXiv:1812.11806, 2018.

[16] Upadhyay S, Faruqui M, Tür G, et al. (Almost) zero-shot cross-lingual spoken language understanding[C]//2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2018: 6034-6038.

[17] Wang M, Deng W. Deep visual domain adaptation: A survey[J]. Neurocomputing, 2018, 312: 135-153.

[18] You K, Long M, Cao Z, et al. Universal domain adaptation[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 2720-2729.

[19] Bapna A, Tür G, Hakkani-Tür D, et al. 2017. Towards zero-shot frame semantic parsing for domain scaling [J]. Proc. Interspeech 2017: 2476-2480.

[20] Lee S, Jha R. Zero-shot adaptive transfer for conversational language understanding[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 6642-6649.

[21] Shah D, Gupta R, Fayazi A, et al. 2019. Robust zero-shot cross-domain slot filling with example values [C] // Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 5484-5490.

[22] Liu Z, Winata G I, Xu P, et al. 2020. Coach: a coarse-to-fine approach for cross-domain slot filling [C] // Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 19.

[23] Wang L, Li X, Liu J, et al. 2021. Bridge to Target Domain by Prototypical Contrastive Learning and Label Confusion: Re-explore Zero-Shot Learning for Slot Filling[C]//Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. 9474-9480.

[24] Takebayashi Y. Spontaneous speech dialogue system TOSBURG II—the user-centered multimodal interface[J]. Systems and computers in Japan, 1995, 26(14): 77-91.

[25] Sordoni A, Galley M, Auli M, et al. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses[C]//HLT-NAACL. 2015.

[26] Yao K, Zweig G, Peng B. Attention with intention for a neural network conversation model[J]. arXiv preprint arXiv:1510.08565, 2015.

[27] Vinyals O, Le Q. A neural conversational model[J]. arXiv preprint arXiv:1506.05869, 2015.

[28] Ham D, Lee J G, Jang Y, et al. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 583-592.

[29] Lin W, Tseng B H, Byrne B. Knowledge-Aware Graph-Enhanced GPT-2 for Dialogue State Tracking[C]//Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. 2021: 7871-7881.

[30] Wu C S, Hoi S C H, Socher R, et al. TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue[C]//Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2020: 917-929.

[31] Zhang H, Liu Z, Xiong C, et al. Grounded Conversation Generation as Guided Traverses in Commonsense Knowledge Graphs[C]//Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020: 2031-2043.

[32] Zhou H, Young T, Huang M, et al. Commonsense knowledge aware conversation generation with graph attention[C]//IJCAI. 2018: 4623-4629.

[33] Zhong P, Wang D, Miao C. Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations[C]//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: 165-176.

[34] Zhang S, Dinan E, Urbanek J, et al. Personalizing Dialogue Agents: I have a dog, do you have pets too?[C]//Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018: 2204-2213.

[35] Xu M, Li P, Yang H, et al. A Neural Topical Expansion Framework for Unstructured Persona-Oriented Dialogue Generation[M]//ECAI 2020. IOS Press, 2020: 2244-2251.

[36] Wu Y, Ma X, Yang D. Personalized response generation via generative split memory network[C]//Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2021: 1956-1970.

[37] Song H, Wang Y, Zhang K, et al. BoB: BERT Over BERT for Training Persona-based Dialogue Models from Limited Personalized Data[C]//Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. 2021: 167-177.

[38] Wen T H, Vandyke D, Mrkšić N, et al. A Network-based End-to-End Trainable Task-oriented Dialogue System[C]//Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers. 2017: 438-449.

[39] Bordes A, Boureau Y L, Weston J. Learning end-to-end goal-oriented dialog[J]. arXiv preprint arXiv:1605.07683, 2016.

[40] Liu C, Zhu S, Zhao Z, et al. Jointly Encoding Word Confusion Network and Dialogue Context with BERT for Spoken Language Understanding[J]. 2020.

[41] Liu Y, Maier W, Minker W, et al. Empathetic Dialogue Generation with Pre-trained RoBERTa-GPT2 and External Knowledge[J]. arXiv preprint arXiv:2109.03004, 2021.

[42] Budzianowski P, Wen T H, Tseng B H, et al. MultiWOZ-A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling[C]//EMNLP. 2018.

[43] Zhu Q, Huang K, Zhang Z, et al. Crosswoz: A large-scale chinese cross-domain task-oriented dialogue dataset[J]. Transactions of the Association for Computational Linguistics, 2020, 8: 281-295.

[44] Henderson M, Thomson B, Williams J D. The second dialog state tracking challenge[C]//Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL). 2014: 263-272.

[45] Tur G, Hakkani-Tür D, Heck L. What is left to be understood in ATIS?[C]//2010 IEEE Spoken Language Technology Workshop. IEEE, 2010: 19-24.

[46] Liu B, Lane I. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling[J]. Interspeech 2016, 2016: 685-689.

[47] Wang Y, Shen Y, Jin H. A Bi-model based RNN Semantic Frame Parsing Model for Intent Detection and Slot Filling[C]//Proceedings of NAACL-HLT. 2018: 309-314.

[48] Haihong E, Niu P, Chen Z, et al. A Novel Bi-directional Interrelated Model for Joint Intent Detection and Slot Filling[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 5467-5471.

[49] Chen Q, Zhuo Z, Wang W. Bert for joint intent classification and slot filling[J]. arXiv preprint arXiv:1902.10909, 2019.

[50] Zue V, Seneff S, Glass J R, et al. JUPlTER: a telephone-based conversational interface for weather information[J]. IEEE Transactions on speech and audio processing, 2000, 8(1): 85-96.

[51] Larsson S, Traum D R. Information state and dialogue management in the TRINDI dialogue move engine toolkit[J]. Natural language engineering, 2000, 6(3-4): 323-340.

[52] Wang Z, Lemon O. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information[C]//Proceedings of the SIGDIAL 2013 Conference. 2013: 423-432.

[53] Lee S. Structured discriminative model for dialog state tracking[C]//Proceedings of the SIGDIAL 2013 Conference. 2013: 442-451.

[54] Lee S, Eskenazi M. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description[C]//Proceedings of the SIGDIAL 2013 Conference. 2013: 414-422.

[55] Sun K, Chen L, Zhu S, et al. The SJTU system for dialog state tracking challenge 2[C]//Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL). 2014: 318-326.

[56] Plátek O, Bělohlávek P, Hudeček V, et al. Recurrent neural networks for dialogue state tracking[J]. arXiv preprint arXiv:1606.08733, 2016.

[57] Kumar A, Ku P, Goyal A, et al. Ma-dst: Multi-attention-based scalable dialog state tracking[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 8107-8114.

[58] Chen L, Lv B, Wang C, et al. Schema-guided multi-domain dialogue state tracking with graph attention neural networks[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 7521-7528.

[59] Lee H, Lee J, Kim T Y. SUMBT: Slot-Utterance Matching for Universal and Scalable Belief Tracking[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 5478-5483.

[60] Schatzmann J, Thomson B, Weilhammer K, et al. Agenda-based user simulation for bootstrapping a POMDP dialogue system[C]//Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers. 2007: 149-152.

[61] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8(3): 229-256.

[62] Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347, 2017.

[63] El Asri L, He J, Suleman K. A Sequence-to-Sequence Model for User Simulation in Spoken Dialogue Systems[J]. Interspeech 2016, 2016: 1151-1155.

[64] Chen W, Chen J, Qin P, et al. Semantically Conditioned Dialog Response Generation via Hierarchical Disentangled Self-Attention[C]//Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019: 3696-3709.

[65] Zhao T, Xie K, Eskenazi M. Rethinking Action Spaces for Reinforcement Learning in End-to-end Dialog Agents with Latent Variable Models[C]//Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1. 2019: 1208-1218.

[66] Pei J, Ren P, Monz C, et al. Retrospective and Prospective Mixture-of-Generators for Task-Oriented Dialogue Response Generation[M]//ECAI 2020. IOS Press, 2020: 2148-2155.

[67] Wen T H, Gasic M, Mrksic N, et al. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems[C]//EMNLP. 2015.

[68] Zhu C, Zeng M, Huang X. Multi-task learning for natural language generation in task-oriented dialogue[C]//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: 1261-1266.

[69] Wang W, Zhang Z, Guo J, et al. Task-oriented dialogue system as natural language generation[C]//Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022: 2698-2703.

[70] Tran V K, Le Nguyen M. Natural Language Generation for Spoken Dialogue System using RNN Encoder-Decoder Networks[C]//Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017). 2017: 442-451.

[71] Guo D, Tur G, Yih W, et al. Joint semantic utterance classification and slot filling with recursive neural networks[C]//2014 IEEE Spoken Language Technology Workshop (SLT). IEEE, 2014: 554-559.

[72] Kurata G, Zhou B, Xiang B, et al. Leveraging sentence-level information with encoder LSTM for semantic slot filling[C]//Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (ACL), 2016.

[73] Palatucci M, Pomerleau D, Hinton G E, et al. Zero-shot learning with semantic output codes[J]. Advances in neural information processing systems, 2009, 22.

[74] Yu X, Aloimonos Y. Attribute-based transfer learning for object categorization with zero/one training example[C]//European conference on computer vision. Springer, Berlin, Heidelberg, 2010: 127-140.

[75] Yazdani M, Henderson J. A model of zero-shot learning of spoken language understanding[C]//Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. 2015: 244-249.

[76] Ferreira E, Jabaian B, Lefevre F. Zero-shot semantic parser for spoken language understanding[C]//Sixteenth Annual Conference of the International Speech Communication Association. 2015.

[77] Xia C, Zhang C, Yan X, et al. Zero-shot User Intent Detection via Capsule Neural Networks[C]//Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018: 3090-3099.

[78] Patrick M K, Adekoya A F, Mighty A A, et al. Capsule networks–a survey[J]. Journal of King Saud University-computer and information sciences, 2022, 34(1): 1295-1310.

[79] Liu H, Zhang X, Fan L, et al. Reconstructing capsule networks for zero-shot intent classification[C]//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: 4799-4809.

[80] Sennrich R, Haddow B, Birch A. Neural Machine Translation of Rare Words with Subword Units[C]//Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2016: 1715-1725.

[81] Luong M T, Sutskever I, Le Q, et al. Addressing the Rare Word Problem in Neural Machine Translation[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. 2015: 11-19.

[82] Jean S, Cho K, Memisevic R, et al. On Using Very Large Target Vocabulary for Neural Machine Translation[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. 2015: 1-10.

[83] He K, Zhang J, Yan Y, et al. Contrastive zero-shot learning for cross-domain slot filling with adversarial attack[C]//Proceedings of the 28th International Conference on Computational Linguistics. 2020: 1461-1467.

[84] Heo S H, Lee W K, Lee J H. mcBERT: Momentum Contrastive Learning with BERT for Zero-Shot Slot Filling[J]. arXiv preprint arXiv:2203.12940, 2022.

[85] Siddique A B, Jamour F, Hristidis V. Linguistically-enriched and context-awarezero-shot slot filling[C]//Proceedings of the Web Conference 2021. 2021: 3279-3290.

[86] Coucke A, Saade A, Ball A, et al. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces[J]. arXiv preprint arXiv:1805.10190, 2018.

[87] Eric M, Goel R, Paul S, et al. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines[C]//Proceedings of the 12th Language Resources and Evaluation Conference. 2020: 422-428.

[88] Han T, Liu X, Takanabu R, et al. MultiWOZ 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation[C]//CCF International Conference on Natural Language Processing and Chinese Computing. Springer, Cham, 2021: 206-218.

[89] Rastogi A, Zang X, Sunkara S, et al. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 8689-8696.

[90] Rastogi A, Zang X, Sunkara S, et al. Schema-guided dialogue state tracking task at dstc8[J]. arXiv preprint arXiv:2002.01359, 2020.

[91] Lei S, Liu S, Sen M, et al. Zero-shot state tracking and user adoption tracking on schema-guided dialogue[C]//Dialog System Technology Challenge Workshop at AAAI. 2020.

[92] Sang E T K, Buchholz S. 2000. Introduction to the CoNLL-2000 shared task chunking [C] // Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop.

[93] Zhong P, Wang D, Miao C. Knowledge-Enriched Transformer for Emotion Detection in Textual Conversations[C]//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: 165-176.

[94] Zhou H, Young T, Huang M, et al. Commonsense knowledge aware conversation generation with graph attention[C]//IJCAI. 2018: 4623-4629.

[95] Van der Maaten L, Hinton G. Visualizing data using t-SNE[J]. Journal of machine learning research, 2008, 9(11).

[96] Bordes A, Usunier N, Garcia-Duran A, et al. Translating embeddings for modeling multi-relational data[J]. Advances in neural information processing systems, 2013, 26.

[97] Wang Z, Zhang J, Feng J, et al. Knowledge graph embedding by translating on hyperplanes[C]//Proceedings of the AAAI conference on artificial intelligence. 2014, 28(1)

[98] Ji G, He S, Xu L, et al. Knowledge graph embedding via dynamic mapping matrix[C]//Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing. 2015: 687-696.

[99] Bengio Y, Ducharme R, Vincent P. A neural probabilistic language model[J]. Advances in neural information processing systems, 2000, 13.

[100] Han X, Cao S, Lv X, et al. Openke: An open toolkit for knowledge embedding[C]//Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations. 2018: 139-144.

[101] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.

[102] Kumar H, Agarwal A, Dasgupta R, et al. Dialogue act sequence labeling using hierarchical encoder with crf[C]//Proceedings of the aaai conference on artificial intelligence. 2018, 32(1).

[103] Bojanowski P, Grave E, Joulin A, et al. Enriching word vectors with subword information[J]. Transactions of the association for computational linguistics, 2017, 5: 135-146.

[104] Joulin A, Grave E, Bojanowski P, et al. Fasttext. zip: Compressing text classification models[J]. arXiv preprint arXiv:1612.03651, 2016.

[105] Takase S, Suzuki J, Nagata M. Character n-gram embeddings to improve RNN language models[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2019, 33(01): 5074-5082.

[106] Reimers N, Gurevych I. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks[J]. arXiv preprint arXiv:1707.06799, 2017.

[107] Zang X, Rastogi A, Sunkara S, et al. MultiWOZ 2.2: A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines[C]//Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI. 2020: 109-117.

[108] Zhang Y, Ou Z, Yu Z. Task-oriented dialog systems that consider multiple appropriate responses under the same context[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(05): 9604-9611.

Publications

Journal

[1] Sixia Li, Shogo Okada, and Jianwu Dang, Intrinsic Representation Mining for Zero-shot Slot Filling, IEICE Transactions, Vol.E105-D,No.11, 2022.

International Conference

[2] Sixia Li and Shogo Okada. An investigation on how zero-shot slot filling benefits the task-oriented dialogue system. The 13th International Workshop on Spoken Dialogue Systems Technology. 2023.

[3] Sixia Li and Jianwu Dang. Zero-shot Domain Adaptation with Inference Relation Paths for Spoken Language Understanding, 2021 Proc. APSIPA.

[4] Sixia Li, Jianwu Dang, and Longbiao Wang. Spoken Language Understanding with Sememe Knowledge as Domain Knowledge, 2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP). IEEE, 2021: 1-5.

[5] Sixia Li and Jianwu Dang. Zero-shot methods of spoken language understanding on mobile transportation-like scenario, Interspeech 2020 Satellite Workshop of Spoken Language Interaction for Mobile Transportation System.

[6] Sixia Li, Shogo Okada, and Jianwu Dang. Interaction Process Label Recognition in Group Discussion, 2019 International Conference on Multimodal Interaction. 2019: 426-434.

[7] Taiyang Guo , Sixia Li, Shogo Okada, and Masashi Unoki. Investigation of Noise-Reverberation-Robustness of Modulation Spectral Features for Speech-Emotion Recognition. 2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC) 2022.

[8] Wenqing Wei, Sixia Li, and Shogo Okada, Investigating the relationship between dialogue and exchange-level impression, 24th ACM International Conference on Multimodal Interaction, 2022.

[9] Wenqing Wei, Sixia Li, Shogo Okada, and Kazunori Komatani. Multimodal user satisfaction recognition for non-task oriented dialogue systems[C]//Proceedings of the 2021 International Conference on Multimodal Interaction. 2021: 586-594.

[10] Xiaohan Shi, Sixia Li, and Jianwu Dang. Dimensional Emotion Prediction Based on Interactive Context in Conversation[C]//INTERSPEECH. 2020: 4193-4197.

Domestic Conference

[11] Sixia Li and Shogo Okada. How zero-shot slot filling benefits the task-oriented dialogue system - An investigation -. SLUD, 2022.

[12] Sixia Li and Jianwu Dang. Pipeline Zero-shot slot filling based on the fusion of context information with slot information and ontology. SLUD, 2021.

[13] Yan Fu, Sixia Li, and Jianwu Dang. Constructing persona dialogue system with common sense knowledge. Acoustic Society of Japan. 2022.

[14] Wenqing Wei, Sixia Li, and Jianwu Dang. A Study of Punctuation Prediction Model for Domain-free Text[C]//IEICE Conferences Archives. The Institute of Electronics, Information and Communication Engineers, 2020.