Doctoral Dissertation

Learning and adaptation of end-to-end multimodal dialog management

Tung The Nguyen

June 25, 2020

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Tung The Nguyen

Thesis Committee:

Professor Satoshi Nakamura	(Supervisor)
Professor Taro Watanabe	(Co-supervisor)
Professor Wolfgang Minker	(Ulm University)
Assistant Professor Koichiro Yoshino	(Co-supervisor)
Associate Professor Sakriani Sakti	(Co-supervisor)

Learning and adaptation of end-to-end multimodal dialog management[*]

Tung The Nguyen

Abstract

Goal-oriented dialog systems are gaining much attention from the machine learning research community, due to their practical applications to various tasks. In the work flow of a dialog system with a user, the management process is critical. Because the dialog management process directly governs the system's behavior, which decides whether the system successfully achieves its goal. In many tasks, non-verbal information from different modalities is essential for the success of the system. As a result, multiple studies have attempted to incorporate such multimodal information into dialog management. However, the current studies generally follow the modular-based approach when creating multimodal dialog systems, and use additional components to process the multimodal features that are needed. Since these components are usually built for a specific task, they cannot be reused in a different dialog system. Thus, systems built using the modular-based approach are difficult to overhaul and adapt to new tasks. In this research, I study the application of the end-to-end approach using neural networks for the dialog management of multimodal goal-oriented systems. Among the vast number of challenges in this topic, my research tackles the following problems:

1. **Multimodal information fusion:** In many tasks, handling multimodal information is important to achieve success. However, the existing works in multimodal dialog management only use the simple concatenation of input

[*]Doctoral Dissertation, Graduate School of Information Science, Nara Institute of Science and Technology, June 25, 2020.

features as the fusion method, which is inefficient, leading to undesired performance. Two main challenges must be faced when considering the multimodal fusion problem: the abstraction level of the input modalities and the interaction features. In this research, to combine features of multimodal input, I propose a neural network-based termed ***Hierarchical Tensor Fusion Network***. This method combines two existing fusion methods: hierarchical fusion and tensor fusion. Experimental results show that the proposed hierarchical tensor fusion network outperforms existing fusion methods in terms of accuracy in the deception detection and sentiment analysis tasks.

2. **Data sparsity:** A popular approach for dialog management is the reinforcement learning (RL) framework that learns a dialog policy that governs the system's action selection procedure. However, the amount of samples needed to learn an optimal solution is usually prohibitive, especially for multimodal dialog tasks, where data sparsity is a huge challenge. In such a situation, policy adaptation is an effective solution, since it allows us to exploit the knowledge from learning a policy in an existing task (the source task) to improve the policy's training in a new task (the target task). Current works in policy adaptation for dialog management use a weight initialization strategy, which requires RL training of the target policy. Although this method is effective, its adaptation process is time-consuming and the learned policy might produce inferior performance due to the small amount of available data. In this paper, I propose a novel approach for dialog policy adaptation called ***Dialog Policy Reuse Algorithm - DPRA*** that does not require training by reinforcement learning on the target task. Experiments show that DPRA greatly reduces training times, and its learned policy outperforms polices trained by existing method when the target task's dataset is limited.

Keywords:

dialog system, dialog management, multimodal processing, reinforcement learning, policy adaptation, mixture density network

Acknowledgements

First and foremost, I would like to express my deepest thanks toward Prof. Satoshi Nakamura for his continuous and enthusiastic supervision during three years of my doctor course. His kind advices and guidance, which helped me realize my weaknesses and improved myself to become not only a better student but also a better researcher. I have been and will always be looking up to him as a great teacher.

Professor Taro Watanabe and Professor Wolfgang Minker; who have been taking part as members of my master thesis's committee, I wish to thank them for their careful reviews as well as many insightful advices to complete my thesis.

I would like to thank Asistant Prof. Koichiro Yoshino for his tireless support that helped me overcame all the obstacles during my research. His knowledge and professionalism really impresed me as well as other members of my lab. I will always be looking at him as a model for me to follow. His guidance and suggestions have helped me a lot in shaping up the idea of this research. In addition to that, his tutorship has equipped me with lots of useful skills that helped me to complete the research in my doctor course.

I wish to send my gratitude to Associate Prof. Sakriani Sakti; her advices for my research were extremely valuable and helped me a lot to complete it with success.

Especially, I want to thank Ms. Manami Matsuda, my lab's assistant, who has always given great helps, both in the lab and in my daily life. I also want to send my thanks to all the lab's members, from whom I received great support.

Finally, I want to send my deepest gratitude to my family. During my doctor course, they have been continuously giving me motivation and inspiration to keep going forward. Without them, this work could not have been completed.

Contents

Acknowledgements **iii**

1. Introduction **1**

1.1 Background 1

1.2 Goal-oriented dialog modeling 3

1.3 Challenges of multimodal dialog management 6

1.3.1 Fusion of multimodal information in dialog systems 7

1.3.2 Data sparsity problem and policy adaptation 9

1.4 Proposed solutions and approaches 10

1.4.1 Multimodal fusion with hierarchical tensor fusion network 10

1.4.2 Improving policy learning with policy adaptation 11

1.4.3 Study on multimodal goal-oriented dialog systems 12

1.5 Thesis overview 13

2. Preliminaries **14**

2.1 Reinforcement learning 14

2.2 Dialog management using reinforcement learning 16

2.3 Policy adaptation 18

2.4 Multimodal dialog tasks 19

2.5 Dialog management performance metrics 22

3. Multimodal fusion with hierarchical tensor fusion network **23**

3.1 Related works 24

3.2 Method 28

3.3 Evaluation I - Deception detection 29

3.3.1 Dataset 29

3.3.2 Features extraction 30

3.3.3 Results of deception detection 31

3.3.4 Relationship between deception detection accuracies and numbers of parameters 33

3.3.5 Dialog management performance based on predicted deception labels 34

3.4 Evaluation II - Sentiment analysis 35

3.4.1 Dataset 35
3.4.2 Feature extraction and training details 36
3.4.3 Results 39
3.5 Conclusion 40

4. Dialog management with multimodal information 41
4.1 Modular-based approach 41
4.2 End-to-end approach 43
4.3 Multimodal user simulation 45
4.4 Dialog management training with policy gradient methods 47
4.5 Evaluation 48
4.5.1 Dataset 48
4.5.2 Observations/Features extraction 49
4.5.3 Results 50

5. Policy reuse with action relation probability 51
5.1 Related works 51
5.2 Method 52
5.2.1 Policy adaptation with action-relation probability 53
5.2.2 Action-relation probability modeling with mixture density network 54
5.3 Evaluation 62
5.3.1 Setting 62
5.3.2 Policy adaptation between similar tasks 63
5.3.3 Policy adaptation between distinctive tasks 69

6. Conclusions and future directions 73
6.1 Conclusions 73
6.2 Future directions 74

References 75

Appendix 83

A. Proof of equation 1 and equation 10 83

B. Results **84**

C. Training dialog policy with REINFORCE **87**

Publication list **89**

List of Figures

1 Example of utilizing multimodal information in goal-oriented dialog systems. 3
2 Example of a modular-based dialog system that handles multimodal information. The input of multimodal processing module can be from visual or acoustic information of the conversation topic or user. 4
3 Example of multimodal end-to-end dialog systems. 6
4 Examples of end-to-end dialog management. 18
5 Flow chart of the health consultation conversation. 20
6 Example of tasks that need processing of multimodal information. 23
7 Early and late fusion network architecture. 25
8 Hierarchical fusion architecture. 25
9 Tensor fusion architecture. 26
10 Hierarchical tensor fusion architecture. 28
11 Effect of network depth on detection performance. 34
12 Hierarchical tensor fusion network for sentiment analysis task with three modalities. 37
13 Hierarchical fusion network for sentiment analysis task with three modalities. 38
14 Modular dialog system for the health consultation task. 42
15 Multimodal dialog management using the end-to-end approach. . 45
16 State transition modeling by mixture density network.. 56
17 Working procedure of DPRA. 59
18 Experiment procedure of policy adaptation evaluation. Each phase is illustrated with different colors and numbering. 63
19 Average reward received per episode of adapted policies in similar tasks setting. 84
20 DA selection accuracy received per episode of adapted policies in similar tasks setting. 85
21 Average reward received per episode of adapted policies in distinctive tasks setting. 86

22 DA selection accuracy received per episode of adapted policies in distinctive tasks setting. 87

List of Tables

1 Dataset partitions. 30
2 Deception detection performance of models based on different fusion methods. Precision, Recall, and F1-score are measured for the positive (Lie) label. 31
3 Accuracy of dialog acts selection when using different deception labels result. 35
4 Detail statistics of the CMU-MOSEI dataset. 36
5 Results of sentiment analysis on the CMU-MOSEI dataset. ↑ indicates metrics that higher is better. ↓ indicates metrics that lower is better. 39
6 Rewards in each turn for all possible combination of (s, a) in the state-action space. 43
7 Statistics of deception labels and dialog acts in the health consultation corpus. 49
8 Accuracy of dialog act selection when using different approaches. . 50
9 Reward definition for the source task. 64
10 Reward definition for the target task. 65
11 Comparison of training time required for different policy adaptation methods. 65
12 Average reward per episode of the learned policies with different amounts of available data. Numbers in bracket indicates 95% confidence interval. 66
13 Dialog act selection accuracy of the learned policies with different amount of available data. Numbers in bracket indicates 95% confidence interval. 67
14 Average reward per episode of the policies that take only use linguistic features (single-modality). Numbers in bracket indicates 95% confidence interval. 68

15 Dialog act selection accuracy of the learned policies that only use linguistic features (single-modality). Numbers in bracket indicates 95% confidence interval. 69
16 Emotion elicitation dataset statistics. 70
17 Average reward per episode of the learned policies with different amount of available data. Numbers in bracket indicates 95% confidence interval. 71
18 Dialog act selection accuracy of the learned policies with different amount of available data. Numbers in bracket indicates 95% confidence interval. 72

1. Introduction

1.1 Background

Goal-oriented dialog systems refer to the type of conversational agents that have a specific objective that must be achieved through conversations with a human user [41, 52]. *Task-oriented* is another term that refers to this specific type of dialog system, since its goal is to complete a certain task. Although dialog systems can be constructed to converse with multiple users (multi-party), this research only addresses single-party dialogs, where the interaction between the system and the user is one-to-one. In contrast to goal-oriented studies, another branch of dialog research tackles about *chat-oriented* or *chat-based* systems, where the system's purpose is to deliver coherent and relevant responses to the conversation topic [51, 38, 39]. Although current studies on chat-oriented dialog systems have achieved promising results, the real-life applications of such system are still limited. As a result, my research focuses on goal-oriented dialog system, which has various applications in practical situations.

In the past, when discussing dialog systems, we usually referred to computer programs. However, with technological advance, the definitions of dialog system and conversational agent have changed. A dialog system can now refer to a smartphone application or an embedded conversation module on a robot. The applications of goal-oriented dialog systems are amazingly diverse. Some systems assist users in such daily tasks as traffic navigation, news summarizing, or booking restaurant. Apple's Siri, Microsoft's Cortana, or car navigation systems are prime examples of such successful, practical dialog systems. Goal-oriented dialog systems can also be used for more active roles, such as managing of discussions or persuading the users to accept recommendations. For example, a commercial seller system might convince users to buy its recommended products or a health consultation system might persuade that persuades its users to adopt a healthier lifestyle.

In the mid 50s of the 20th century, the interactions between humans and computers or machines are text-based. Traditional dialog systems were only built for handling lexical input (text) from user. As technologies steadily advances, smart devices are able to capture non-verbal information such as facial expressions, movements, or changes in the human vocal pitch. In many tasks, such information is important for the system to successfully reach its goal. For example, assume that an automatic health consultation system has just suggested that a user start running three times a week. Naturally, most users do not immediately jump off the coach and run a marathon; they ponder, consider, and reflect. During this period, if the system has information about the user's facial expressions, it can evaluate the user's interest level in the recommendation and determine whether to continue to persuade the user or to recommend a new activity instead. For some tasks, the multimodal information that used by the system might come from the conversation topic. Assume that a commercial agent system is recommending the purchase of a certain car (Figure 1). The system has access to the car's image, but no information about the available color. The system wants to compliment the car's color. In such a situation, it needs to process the car's image to extract a color and responses: *"Red looks really great on this car, doesn't it?"*. Such a response is more persuasive and helps the system to successfully reach its goal. These are just a couple examples of the effectiveness of multimodal information in goal-oriented dialog systems.

Based on the above reasons, the incorporation of multimodal information has become a popular topic in the dialog research community. In general, three main modalities have been commonly used in previous studies:*linguistic*(text), *visual*, and *acoustic*. Note that multimodal processing also considers other modalities including brain signals (EEG or fMRI) and biological readings (heart rate, blood pressure). In this research, I avoid using these modalities because human mainly perceives information from visual, acoustic, and linguistic features when talking with each other and solving a task related to the conversation. Since the goal of dialog research is to create human-like, intelligent conversational agents, I follow previous works and focus on visual, acoustic, and linguistic modalities. The schemes that takes input from two or more modalities are called *multimodal dialog system*. They are the scope of this study. My research focuses on the management process of multimodal goal-oriented dialog systems.

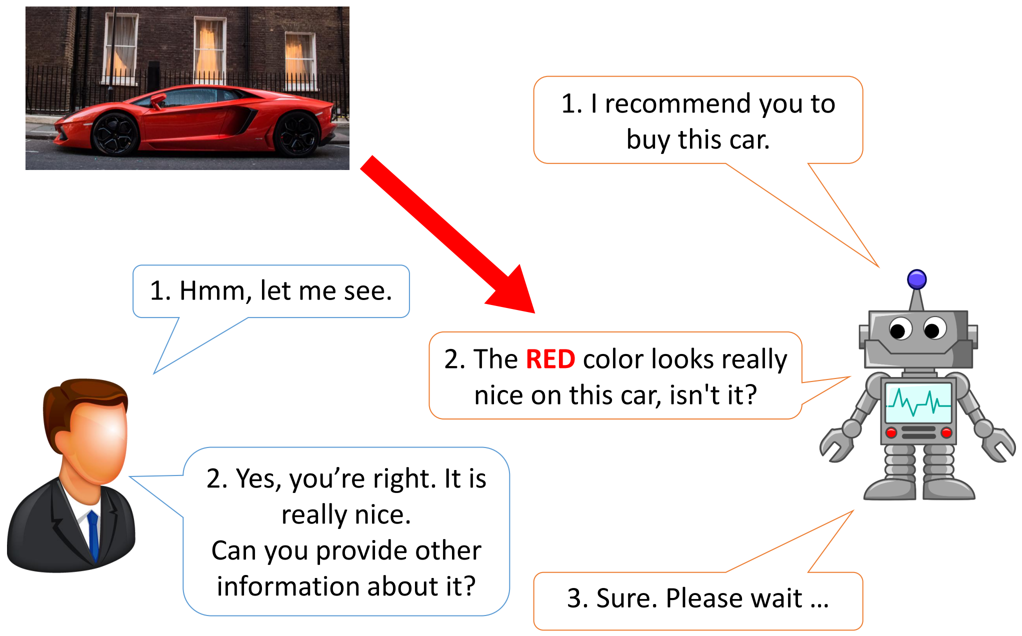
1. I recommend you to buy this car.
1. Hmm, let me see.
2. The RED color looks really nice on this car, isn't it?
2. Yes, you're right. It is really nice. Can you provide other information about it?
3. Sure. Please wait ...

Figure 1. Example of utilizing multimodal information in goal-oriented dialog systems.

1.2 Goal-oriented dialog modeling

This section discusses the general approaches for modeling multimodal goal-oriented dialog. In principle, the interactions between systems and users can be divided into smaller units, which is called *turn*. A modular dialog system contains several components, each of which handles a small part of the interaction process between the system and the user every turn (Figure 2).

The following is the working process of the dialog system. The *Natural Language Understanding (NLU)* module takes input as the user's spoken utterance and recognizes the user's dialog act (DA), which is a special label that represents the user's intention, or what he wants to say. The *Multimodal Processing* module takes the input multimodal information and generates a multimodal label, which depends on the task's goal. For example, a commercial agent may need a label

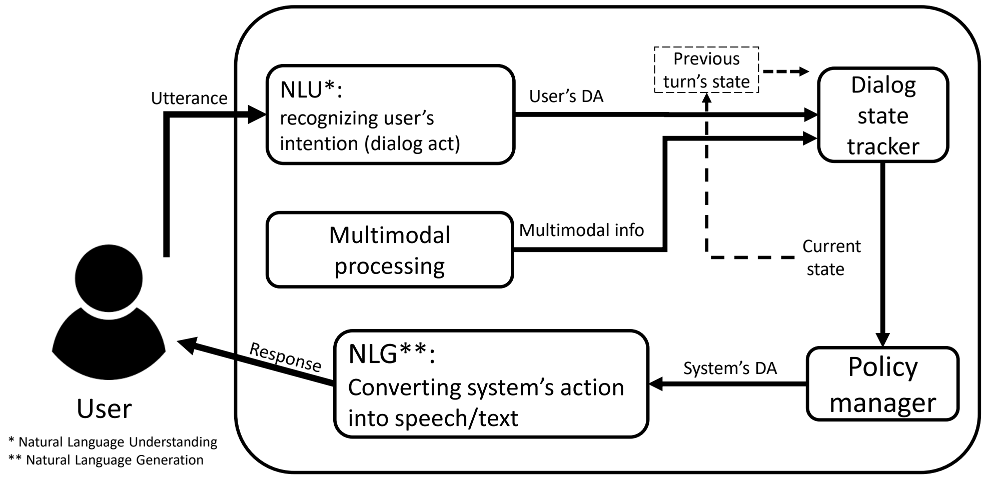

Figure 2. Example of a modular-based dialog system that handles multimodal information. The input of multimodal processing module can be from visual or acoustic information of the conversation topic or user.

that represents the user's interest level in the recommended product. In the emotion elicitation task, where the system tries to improve the user's mood, this label should represent the user's emotion level. As mentioned in Section 1.1, the source of the input multimodal information also varies by the task. Figure 2 shows an example where the Multimodal processing module takes input information from the user. Next, the *Dialog State Tracker (DST)* module takes the multimodal label and the user's DA as input and generates a dialog state, which summarizes important information of the current turn. The DST's working process is recurrent, since the previous turns are also considered during dialog state tracking. The resultant dialog state from the DST is then fed to the *Policy Manager* or the *Dialog Manager* module. This module performs dialog management process, which selects an appropriate system's dialog action as a response to the user. The system's action represents the functions provided by the system provides in conversations with the user. For example, in hotel booking tasks, the system action set can contain actions such as *inform(hotel_name)* or *request(price_range)*. The policy manager can be built by hand-crafted rules or be trained using supervised learning methods with manually annotated system action labels provided by experts. However, the most popular approach for dialog management utilizes the *reinforcement learning (RL)* framework, which trains a *dialog policy* that governs the action selection procedure. Finally, the *Natural Language Generation* module creates a sentence in natural language as a response to the user. The generated sentences can be text or spoken utterances, depending on how the system is designed. After receiving the system's response, the user makes a new utterance, which starts a new dialog turn. This interaction process between the system and the user continues until the conversation ends. Modular-based systems currently remains a common approach for building goal-oriented dialog systems.

The major weakness of modular-based dialog systems is their rigidity. Since each task requires a different kind of multimodal labels, except for NLU and NLG, which can be used for multiple tasks, the remaining components are mostly task-specific and cannot be reused. This increases the burden if we are going to build multiple dialog systems. For a chat-based dialog system with an end-to-end approach, this problem does not happen, since we can build a system using neural network models such as encoder-decoder and adoption to a new domain can be done immediately by training on the new data. Motivated by this observation, several studies introduced neural networks for modeling goal-oriented dialogs, allowing the system to be trained in an end-to-end manner. We call such systems built using this approach *end-to-end* dialog system. The following two works, [59, 11] are among the first that constructed goal-oriented dialog systems using the end-to-end approach .

Figure 3 shows an example of the structure of a dialog system built with the end-to-end approach. In principle, the roles of natural language understanding, dialog state tracking, and dialog management are all performed by a single component: the dialog manager. In practice, this can be done using neural networks to construct each components and fully connect them. This allows us to train the dialog management process in an end-to-end manner by propagation from the output actions back into the input layer of multimodal information. In this case, our system has an *end-to-end dialog management* procedure. As shown in Figure 3, the NLG is not connected to the policy manager; it is trained separately. The

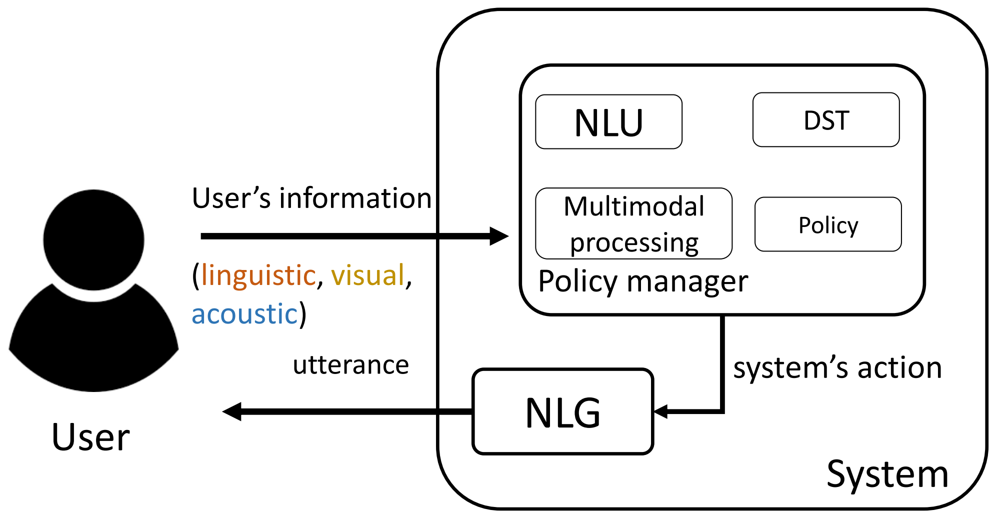

Figure 3. Example of multimodal end-to-end dialog systems.

example in Figure 3 denotes single-modality system, where the user's utterance is used as input. To the best of my knowledge, most current works on end-to-end goal-oriented dialog systems are for single-modality dialogs, The number of studies of this approach for multimodal dialog systems is still very limited.

It is also possible to make a completely end-to-end goal-oriented dialog system that generates natural sentences [13]. This approach resembles the method usually used in chat-based dialog systems [44, 26]. However, with this method, we do not train a policy for dialog management. Instead, we train a system that learns to imitate the example sentences prepared by humans; in other words, supervised learning. As a result, the system's performance is bounded by the quality of the dataset. For complex tasks,since this supervised-learning-based approach usually provides inadequate results, it is beyond the scope of my research, where RL is used for dialog management.

1.3 Challenges of multimodal dialog management

In this section, I discuss issues toward the realization of end-to-end dialog management for multimodal dialogs, as described in the previous section. After scrutinizing the current works in this topic, I identify two major challenges that have not been fully addressed in existing studies:

1. **Multimodal fusion:** Current works on multimodal dialogs use simple concatenation to combine multimodal information, a strategy that is inefficient. This fusion method is widely used in both goal-oriented and chat-based dialog systems. I propose a novel method for multimodal fusion using a neural network.

2. **Data sparsity:** Data scarcity has always been a huge challenge for goal-oriented dialog research. Especially for multimodal dialogs, the current available corpora are very limited. Existing works have tackled this problem using policy adaptation, which allows the utilization of knowledge when learning a policy from a source task to improve the policy training in a target task. However, these works follow a *weight initialization* strategy, which is very time-consuming and results in the learned policy with low performance when the data is insufficient. I propose a new policy adaptation method that does not require RL training of the target task's policy, thus solving the problems in current works.

While the topic of this thesis focuses on end-to-end dialog management for multimodal goal-oriented dialogs, both proposed solutions for these issues can be applied to a broader field without any restrictions.

1.3.1 Fusion of multimodal information in dialog systems

Multimodal dialog systems are a burgeoning field of interdisciplinary research that attracts both the vision and language communities due to the potential applications. This section discusses multimodal goal-oriented dialog systems, where RL is used for dialog management. However, some works on systems with supervised learning-based methods are also mentioned, to deepen the context for the readers.

Visual question answering (VQA) [8], which is the most popular task setting, is commonly used in multimodal dialogs. In this task, a conversational agent answers questions regarding an assigned image based on a dialog history. Another work [9] proposed a visual dialog system to handle this task by using reinforcement learning. Separate encoders are used to learn embeddings that represent the dialog context (linguistic) and the given image. The linguistic and visual embeddings are combined using concatenation and fed into a history encoder, which actually performs the dialog state tracking. In a similar approach, two works [60] and [58] fused the visual and linguistic information together by concatenating the embedding vectors. Such a fusion method is also known as *early fusion* or *feature-level fusion* in other literatures.

The use of acoustic information in goal-oriented dialogs has also been investigated. A work [17] extended the visual question answering task into multimodal scene scenario in which the system answers the user's questions regarding a given video with both visual and acoustic information. Similar to [9], this work uses encoders to learn the embeddings of visual, linguistic, and acoustic information. Instead of simple naive fusion like in previous works, [17] used the *attentional multimodal fusion*, with different attention weights when combining visual and acoustic embeddings. After that, the resultant vector is concatenated with the embedding of the current user's question and the dialog history embedding. Finally, this combination is fed into the decoder to generate a full-sentenced answer to the user. This fusion approach provides good improvement since it focus on the specific modalities of input based on the decoder's current state to predict the word sequence in the video description of the output answer.

Other research [40] proposed a multimodal dialog system for handling the therapy of autistic children. In addition to the visual and acoustic information, the features captured by a wristband worn by the children (users) were also used. This work uses recurrent neural network (RNN) with long short-term memory cell the decoder to separately classify each input modality into labels of engagement level. The final decision of engagement label is decided by majority vote of the labels from each modality. This label is then used for the dialog management process. With this fusion method, since the multimodal information is combined in the majority vote phase, it is termed *late fusion* or *decision-level fusion.*

In summary, the current works in multimodal dialogs rely heavily on concatenation for the fusion of different information modalities. Although such more advanced methods as attentional fusion or late fusion have been considered, even these methods contain drawbacks, as explained in more detail in Section 3.1.

1.3.2 Data sparsity problem and policy adaptation

The reinforcement learning framework is a popular method for goal-oriented dialog management. In principle, this framework provides mechanisms to learn solutions for any task with sparse signals called rewards; in other words, human supervision is not necessary in RL training. Unfortunately, the amount of samples needed to learn a optimal solution is usually prohibitive, especially in multimodal dialogs, where data sparsity is a huge challenge [43, 33, 26]. Policy adaptation, or policy transfer, is a very useful technique that can tackle this problem in reinforcement learning.

Policy adaptation refers to the process of reusing the knowledge that was learned in one or multiple source tasks to a new target task. Various research has studied about it in RL and proposed different techniques whose results are very promising including the acceleration of convergence rates and the reduction of data volume requirements [48, 23, 4, 20].

Within the scope of reinforcement learning-based dialog management, the application of policy adaptation remains very limited. Current works in dialog policy adaptation [6, 19, 29] follow the weight initialization strategy, which is composed of two steps: pre-training and fine-tuning. The pre-training process trains a policy in the source task, where the policy is usually represented by a neural network. A part of the source policy's weight parameters is used for the initialization of the neural network's weights in the training of a policy in the target task.

However, when we do not have enough data for the target task, this strategy does not work well because it barely uses the knowledge from the source task's policy and the target task's policy is basically trained from scratch. Consider a situation where we have a dialog policy that handles restaurant reservations, which we want to adapt to manage the hotel reservation task. Obviously, we expect that this adaptation can be performed with a minor adjustment to the policy of the restaurant reservation task. Existing works on dialog policy adaptation do not satisfy this requirement; their adaptation process is very time-consuming. In addition, since the RL training process of dialog policies usually involves in a user simulation, if we do not have enough data for creating a good simulation, the performance of the learned policies will be inadequate.

1.4 Proposed solutions and approaches

This section briefly discusses about the methods proposed in my research that address the problems in the above discussions.

1.4.1 Multimodal fusion with hierarchical tensor fusion network

The two major challenges of multimodal fusion are balancing the abstraction level of information from different modalities and learning of the feature interactions.

Tian et al. [49] argued that the information from modalities has a different level of abstraction or describe the data at various timescales. A modality's abstraction level can be judged by how easily the task can be solved using the information from this modality. For some tasks, visual information may be more crucial than acoustic information; the reverse phenomenon can be observed in other tasks. A previously proposed abstraction level by [49] resembles the representation level in neural network studies. To balance the difference across modalities, *hierarchical fusion* was proposed by [49], which is done by inputting features from different modalities into different layers of a neural network. This configuration balances the differences and improves the multimodal fusion effectiveness.

Other than the abstraction level of the modalities, the interactions among features are another important aspect when considering multimodal fusion. A previous work [56] suggested that when combining multiple modalities, a neural network needs to learn both intra-modality and inter-modality feature interactions. The former refers to the interactions that happen among features from the same modality. Similarly, inter-modality interactions refer to the interactions between features from different modalities. In early fusion, the interactions are learned simultaneously, which is complex and inefficient. On the other hand, with late fusion, inter-modality interactions are ignored. [56] proposed the *tensor fusion* method that combines embedding features from different modalities with the tensor (outer) product and separates the learning of the intra- and inter-modality interactions.

In the context of multimodal dialog systems, most current works use early fusion methods [9] that cannot balance the modality difference and the feature interaction learning is entangled and inefficient. The late fusion method cannot model the learning of inter-modality feature interactions since the combination is performed at the decision level. A work [17] used attention to fuse modalities allowing the difference of modalities to be balanced in a similar way to hierarchical fusion. However, neither of these methods tackles the problem of feature interactions. Although tensor fusion efficiently learns feature interactions , it cannot address the issue of modality balancing. In conclusion, with multimodal dialogs and other multimodal processing tasks, there is no previous work that has simultaneously addressed both the problems of modality balancing and feature interaction learning.

Motivated by these observations, I propose the hierarchical tensor fusion network, which is a fusion method based on neural networks. It exploits advantages from both the hierarchical and tensor fusion methods and can simultaneously solve both the problem of abstraction level and feature interactions. I hypothesize that hierarchical tensor fusion will allow us to train a better model for multimodal processing tasks. In Sections 3.3 and 3.4, I show empirical results to that prove the proposed fusion method outperforms the existing works in deception detection and sentiment analysis tasks.

1.4.2 Improving policy learning with policy adaptation

As described in the previous section, current policy adaptation in dialog management requires RL training for the policy on the target task. The weights of the target task's policy neural network are initialized using those trained in a source task. Such a weight initialization method improves the learning of the target task's policy. However, for cases where dialog samples are insufficient, this strategy struggles.

In this research, I show that a policy for the target task can be learned without RL on this task. Instead of training a policy by interactions with a user simulator, my proposed method establishes a connection between the policies of the source and the target tasks through a special mapping distribution that is called the *action-relation probability.* The proposed adaptation method, DPRA, learns this distribution from the dialog samples in both tasks and immediately derives a policy for the target task. Thus, DPRA remarkably reduces the learning time. Since the proposed adaptation method does not require the user simulator, the problem of low performance, which is caused by errors in constructing the user simulator, can be avoided.

1.4.3 Study on multimodal goal-oriented dialog systems

In addition to the above two contributions, I also conducted a study on dialog management for multimodal goal-oriented dialog systems using a health consultation task as a use case. A more detailed description of this task is given in Chapter 2. Several works have been conducted about multimodal goal-oriented dialog systems [8, 17, 47, 58]. However, all of these works use a question-answering setting where the dialog system simply extracts the information of an image (visual) or a video scene (visual-acoustic) and answers the user's question. My research is among the first that studies a multimodal dialog system where the user's multimodal information (e.g., facial expressions, vocal pitch, and tone) is considered in the dialog management process. I believe that this setting is more realistic and practical than the popular question-answering task that is currently being used.

In this study, I built two dialog managers using modular-based and end-to-end approaches. Although the application of end-to-end approaches in multimodal dialog management has been investigated in previous studies, these works only used simple concatenation for multimodal fusion [58] or trained the dialog manager using supervised learning methods [17]. On the other hand, I propose an end-to-end dialog manager that utilizes my novel hierarchical tensor fusion for combining information from different modalities. This dialog model is one of the first to incorporate multimodal fusion with end-to-end reinforcement learning-based dialog management. A detailed analysis of the advantages and disadvantages of the dialog managers in this study is provided in Chapter 4. In addition, I experimentally evaluated the performance of these two dialog managers in a health consultation task.

1.5 Thesis overview

The remaining parts of this thesis are organized as follows.

Chapter 2 provides background knowledge in reinforcement learning and dialog management with RL to deepen the readers' comprehension of the proposed methods. In chapter 3, I scrutiny the existing multimodal fusion methods and their disadvantages and explain the hierarchical tensor fusion network, which is my proposed method in this thesis. This chapter also describes how the current problems of multimodal fusion are addressed using the proposed hierarchical tensor fusion method. To evaluate its efficiency, I performed experiments with the deception detection and sentiment analysis tasks.

In chapter 4, I first give an example of a modular-based dialog system that handles multimodal information in a health consultation task. Next, I discuss how to apply end-to-end dialog modeling for the same task, and provide details about how the hierarchical tensor fusion is incorporated into this end-to-end dialog model. In addition, a comparison of the performance of the modular-based and end-to-end approaches is provided in the experiment section.

Chapter 5 explains the proposed dialog policy reuse algorithm. I show how to create a connection that allows us to use the source task's policy for the action selection in the target task. Details explain how the action-relation probabilities are modeled using a mixture density network. I show the results in terms of the performance of the learned polices in comparison with previous works in two setting: adaptation between similar and distinctive tasks.

Finally, in chapter 6, I summarize the works of this research and discusses of the proposed methods and future directions that flow from my work.

2. Preliminaries

This section provides background knowledge and important definitions in reinforcement learning and dialog management that are used in this thesis.

2.1 Reinforcement learning

Reinforcement learning is a popular framework for learning autonomous behavior. I consider the standard reinforcement learning setting where an agent interacts with a environment, a Markov Decision Process (MDP), over a number of discrete time steps [2]. At each time step t, the state, action, and reward are denoted by $s_t \in S$, $a_t \in A$, and $r_t \in R$ respectively. The dynamics of the task, or the environment, are characterized by two random variables. The first one is the state transition probabilities,

$$P^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a). \tag{1}$$

The second environment dynamic variable is the expected reward, which is given by,

$$\begin{aligned} R^a_s &= E[r_{t+1} | s_t = s, a_t = a] \\ &= \sum_{r_{t+1}} r_{t+1} P(r_{t+1} | s_t = s, a_t = a). \end{aligned} \tag{2}$$

In this thesis, I follow the notations by [45], with the reward for selecting action a_t given the state s_t is denoted by r_{t+1} instead of r_t in some other literatures. This notation allows us to emphasize on the fact that both the reward r_{t+1} and the next state s_{t+1} are results of selecting action a_t in the current time step.

The agent's procedure of selecting an action a given a state s is the agent's policy, denoted by $\pi(s, a) = P(a|s)$. We define the *return*, which is the total rewards that the agent receives, as follows,

$$\begin{aligned} R_t &= r_{t+1} + r_{t+2} + \cdots + r_T \\ &= \sum_{k=1}^{T-k} r_{t+k}. \end{aligned} \tag{3}$$

with T is the final time step. The agent's objective is to maximize the expected return $E[R_t | s_t, \pi]$ at each time step t when following a certain policy π. The policy that satisfies this maximum condition is called the *optimal policy*, which is denoted by $\pi^*(s,a)$. To relax mathematical notation burden in the formulas, the policy and optimal policy are also denoted by π and π^* respectively in this thesis. If the agent-environment interactions do not stop, T goes to ∞, we say that our task is *continuing*. If the interactions eventually end when reaching a certain terminal state, then our task is called *episodic*. In this setting, the interactions from the beginning until the agent reaches a terminal state is called an *episode*. The appropriate setting should be chosen depends on the problem we want to solve using reinforcement learning. When our task is continuing, the return becomes infinity and maximizing the objective is impossible. Therefore, in continuing tasks, we use the *discounted return* as the agent's objective, $\gamma \in [0,1)$ is the *discount factor*,

$$R_t = \sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k}. \tag{4}$$

In this case, we say that our task is under the *discounted* setting. Similarly, if the agent's objective is the return, then the setting of our task is *undiscounted*.

Given a policy π, the state-action value is defined as the expectation of the return if action a is chosen at time step t, which is given by,

$$Q^{\pi}(s,a) = E[\sum_{k=1}^{T-k} r_{t+k} | s_t = s, a_t = a, \pi]. \tag{5}$$

$Q^{\pi}(s,a)$ is also referred to as Q-value in some other works. In this thesis, both of these terms are use interchangeably. Similarly, we define a state value of policy π as:

$$V^{\pi}(s) = E[\sum_{k=1}^{T-k} r_{t+k} | s_t = s, \pi]. \tag{6}$$

Please note that the above definitions of state-action, and state value are under the *undiscounted* setting. The state-action and state value for the discounted setting are deduced by replacing R_t with the discounted return at time step t.

There are two classes of reinforcement learning algorithms: value-based and policy-based. In value-based reinforcement learning methods, we estimate the action-state $Q^{\pi}(s,a)$ or the state value $V^{\pi}(s)$ by using a function approximator, such as neural networks or simple value tables. The classic Q-learning [50] or deep

Q-network [30] are examples of this class of algorithms. In contrast to value-based methods, policy-based RL algorithms parameterize the policy π by parameters θ. We update the parameters θ by performing, typically approximate, gradient ascent on the objective of maximizing $E[R_t|\pi]$. There are various policy-based reinforcement learning studies, especially with policy gradient methods, such as Actor-Critic [7, 22, 3] or REINFORCE [53], which is the method that I am using in the evaluations in the remaining parts of this thesis.

The description above explains the fundamentals of the reinforcement learning framework and Markov Decision Process. In MDP formulation, we assume that all the information contained in the state s is fully observable. However, this condition is difficult to fulfill since feature extraction may contain errors that are propagated to the agent's learning process. To alleviate this problem, we usually use the *Partially Observable Markov Decision Process - POMDP*, which can consider the imperfection of state information. In POMDP, the agent decides the most appropriate action based on a *belief* b, which is the distribution over all possible states.

$$b = (P(s_1), P(s_2), \cdots, P(s_n)) \quad S = \{s_1, s_2, \cdots, s_n\} \tag{7}$$

Under the assumption that the belief is Markovian, we can treat the belief as a continuous state with n dimensions and apply RL algorithms to train the agent's policy. Note that if the original state s is continuous i.e., the state space S is infinite, we can use discretization to create a state space with a finite number of elements and apply POMDP formulation above.

2.2 Dialog management using reinforcement learning

We can divide a dialog into multiple turns. Each turn contains a user utterance and a system response. We can formulate the problem of dialog management as an MDP and apply any RL algorithm for solving it. We can define a set of actions for the system to interact with the user and define a reward function based its goal. The set of system actions is defined based on the function of the system's functions in the conversation. The system's action is also referred to by the term *dialog act* or DA. In dialog management by reinforcement learning, we can use the two terms "agent" and "system" interchangeably because the learned policy represents the system's behavior. Since all dialogs only have a finite number of turns or time steps, we need to use the episodic and undiscounted setting as the formulation of dialog management.

At each time step t, the information that is necessary for the action selection procedure is defined as the *observation*, denoted by o_t. The type of information included in the observation depends entirely on the task we are trying to solve. For example, in dialog management, the observation may consist of recognition results of slot information [19, 11, 6], user's dialog action [15, 55], or some high-level multimodal information such as user's deception [34, 32]. Let us recall that the policy is a conditional probability of selecting action a given the state s, $P(a|s)$; thus, the state must include important information for making the decision. A natural approach is to represent the state by the vector of observation or concatenation of observations from multiple time steps. We call this way of representing the state as *explicit state representation.* In modular-based dialog systems [41, 52, 15, 55, 34], the state is represented by this explicit representation.

In the dialog management procedure, the system needs to consider the dialog history, which contains the user's utterances and the system's actions from the beginning. With such a long-term tracking requirement, the explicit representation becomes unsuitable if the observation is high-dimensional or the conversations are lengthy. A common solution is to use a Recurrent Neural Network (RNN) to learn a state embedding, which we can see as a *latent state representation* that stores the dialog history and current observation [59, 11]. Such representation of the state is used in end-to-end dialog modeling [59]. Figure 4 shows an example of this approach. We can see that the RNN plays the role of DST, and its output, the latent state representation, is used by the dialog policy module for management. With the end-to-end approach, we are free from designing a complicated explicit representation of the dialog states.

RL training requires a considerable amount of agent-environment interactions in order to learn a good policy. In dialog management, it is impossible to collect enough dialog samples to fulfill this requirement. [41] propose the use of *user simulator* as a replacement for the real human user, for training the policy. This approach has become the standard for training dialog policies with RL. The user simulator is built from the dialog samples, by maximum likelihood or supervised

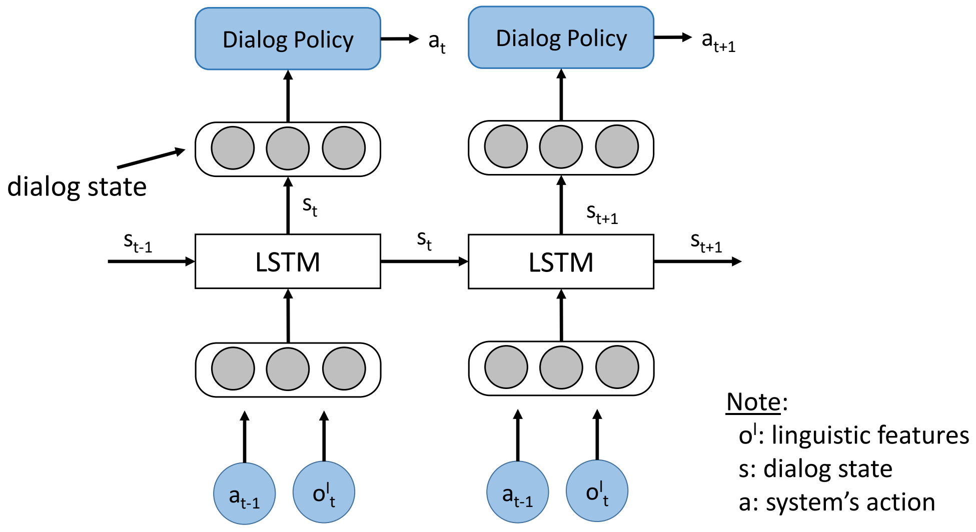

Figure 4. Examples of end-to-end dialog management.

learning methods to imitate the human user's behavior. We can view the user simulator can also be viewed as an approximation of the dialog management task's actual state transition $P^a_{ss'}$, which is provided by the real human user.

2.3 Policy adaptation

Humans can learn a task better and faster by transferring the knowledge retained from solving similar tasks. This observation motivates the idea of transferring knowledge across different but related tasks to improve the performance of machine learning (ML) algorithms. The techniques that allow such knowledge transferring is termed *transfer learning*.

From the middle of the 2000s, application of transfer learning to RL algorithms started to gain much attention, "policy adaptation" adapts the policy trained on the source task to a new task (target task). In principle, the reinforcement learning framework provides mechanisms to learn solutions for any task with sparse signals called rewards; in other words, human supervision is not necessary for RL training. However, the amount of samples needed to learn an optimal solution is usually prohibitive, especially in dialog management, where data sparsity is a substantial challenge [43]. Transfer learning can build prior information from the knowledge collected to solve a set of source tasks and use them to learn a policy in the target task.

We can use many types of knowledge in transfer learning in RL, such as samples, representation, or parameters [23]. *Policy adaptation*, or *policy transfer*, refers to the transfer learning methods that use knowledge of the policy from the source task for the transferring process. We can say that the policy adaptation methods are a subclass of transfer learning solutions in reinforcement learning.

We usually face the problem of data sparsity when training policies for dialog tasks. In this situation, the policy adaptation approach is a promising solution. Multiple studies investigate the application of policy adaptation in reinforcement learning-based dialog management [6, 19, 29]. All these methods show improvement in terms of learning speed, reduction of data requirement, or performance.

2.4 Multimodal dialog tasks

This study uses the health consultation conversation as a use case for multimodal dialog management. With this type of dialog, the system plays the role of a doctor and tries to convince the user to adopt more healthy living habits. The topics of this conversation can be one of the following: sleeping, food, working, exercises, social media usage, and leisure activities. The user can use some deceptive reasons to claim that changing to the new living habits is too difficult for him. Therefore, the system needs to identify the user's deception and provides appropriate action. The system needs to consider the user's facial expressions and other acoustic clues, such as pitch, power or tones, to detect whether the user is lying or not,. Therefore, the task of health consultation is a multimodal dialog management problem. The conversation between the system and the user can be regarded as a negotiation since the system can adjust its recommendation according to whether it believes the user's reason is honest.

Figure 5 shows the desired dialog behavior of the system, which considers the user deceptions. The rectangles in this flowchart indicate system actions. The set of dialog actions for the system include:

Offer: The system suggests the user should change to a new habit.

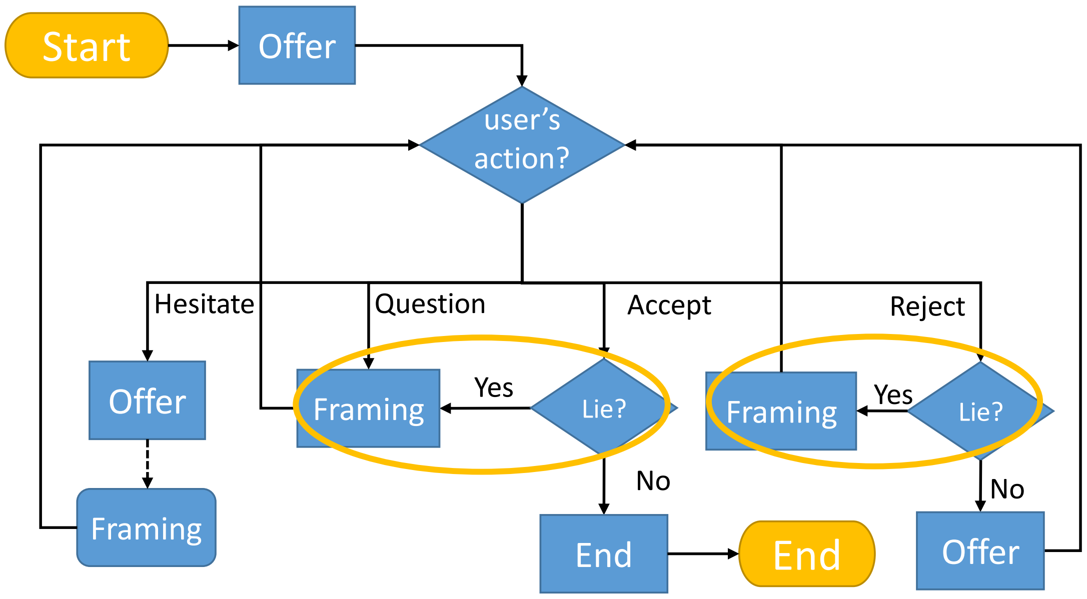

Figure 5. Flow chart of the health consultation conversation.

Framing: The system provides arguments to persuade the user.

End: The system ends the conversation.

Similar to the work in [15], this study uses *Framing* to represent the system's dialog action that is used for persuading the user. In particular, this action represents the function that the system gives information about a new habit (system's recommendation), the health benefits of the new habit, and the negative effects of the user's current habits. In the health consultation task, there are two situations where the user can tell a lie. The first situation happens when the user uses some deceptive reasons to reject the system's recommendation. In the second situation, the user pretends to accept that he/she will change to the suggested habit. In both cases, the expected action that the system should select is Framing. The action set of the user contains the following four dialog actions:

Accept: The user agrees to change the habit.

Reject: The user gives reasons why they cannot change their habit.

Hesitate: The user says he/she is unsure about whether to accept the offer or not.

Question: The user asks the system to provide more details about the new habit.

Below is an example of a conversation in the health consultation task, with the numbering indicates turn number.

System-1: Hello, thank you for coming today. Let's talk about your working habits first. I think that you are spending too much time sitting continuously during working and you should take breaks more often. I suggest that you take a break every hour, and during breaks, you should stand up, move away from your desk, and do some light stretches outside of your office. Offer
User-1: Why should I move away from my desk during breaks? I feel more comfortable and relaxed sitting in my chair, reading news than going outside. Question Truth
System-2: It may be more comfortable for you to do so, but if you still work with a computer like that during breaks, your eyes cannot rest. Continuously straining your eyes like that is not good at all and worsen your eye health. Framing
User-2: I see that taking breaks is good for my eyes, however, if I go out to take a break, I feel disrupted, and my productivity will be reduced when return. I prefer to take breaks at my desk. Reject Deception
System-3: Maybe your productivity will be reduced a little bit; however, in return, your health will be better in the long term. Framing
User-3: Okay, I'll try to do as you suggested. Accept Deception
System-4: That's great! In case you also want to know, sitting continuously for a long time also puts a lot of stress on your spine and neck, causing many severe health problems. Doing some light stretches help you regain posture and release stress from your body. Framing
User-4: Oh, I see. Sometimes, I also feel my neck hurt a bit while doing work, too. Maybe your suggestion can help. Hesitate Truth
System-5: If you find it too distracted for your work to go out every break, how about going out and do stretches just once or twice a day? For example,

you can take 3-4 breaks per day, and during one of those, you do some stretches; for the remaining breaks, you stay at your desk. Offer
User-5: That sounds good to me. Okay then, I will give it a try. Accept Truth
System-6: That's right. If you follow my recommendation, soon, you will see the problems related to your neck gone in no time! End

2.5 Dialog management performance metrics

This section explains the two common metrics used to measure dialog management performance in this thesis.

The first metric that I use is the *average reward per episode*, also referred to as *total expected return* [53, 30]. It is measured by summing all the rewards in each episode and average the result over all the evaluation episodes. The average reward per episode is the standard performance metric that is being used in most of reinforcement learning research, especially for the episodic setting. Since the dialog manager's policies that are evaluated in this research are built by using reinforcement learning, this is a suitable metric to evaluate the dialog managers' performance. In reinforcement learning tasks, such as autonomous control, it is common to use the same environments to train and test the policies. However, in reinforcement learning-based dialog management, sometimes different environments (the user simulators) are used in training and testing. Since goal-oriented dialog corpora are usually limited, this strategy ensures that the learned policies for dialog managers truly have good performance and are robust to different datasets.

The second performance metric is *dialog act (DA) selection accuracy*, a popular measurement in various dialog studies. With this metric, we measure the dialog managers' performance in a similarly to classification, by treating the system's action as labels. In practice, human experts manually annotate the dataset with the most appropriate system's actions. These actions are used as ground truth for measuring the dialog manager's action selection accuracy. Usually, we need to conduct an agreement analysis to ensure that the system action annotation is similar across different annotators.

3. Multimodal fusion with hierarchical tensor fusion network

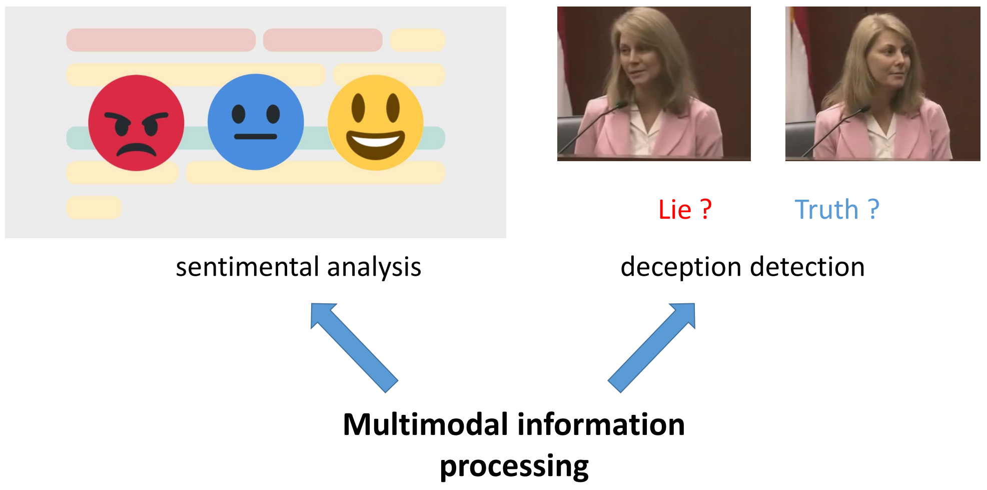

Figure 6. Example of tasks that need processing of multimodal information.

With the recent rapid progress of technologies, the interaction between humans and machines has become multimodal. Therefore, multimodal information processing problem has attracted much attention from the machine learning community. This chapter discusses integrating information from multiple modalities and the solution proposed in this research, the hierarchical tensor fusion. First, I analyze the currently popular methods for multimodal fusion. From this analysis, I identify two main problems that we need to address: balancing the modalities and feature interactions learning. Next, I give the details of how the proposal, hierarchical tensor fusion, can solve these problems. Finally, I show results and analysis of the experiments using different fusion methods on deception detection and sentiment analysis tasks.

3.1 Related works

This section describes current methods of modality combination using an example of the deception detection task. The neural network model I used in this study is the multi-layer perceptron (MLP), a fully-connected feed-forward neural network with multiple hidden layers. The output contains two neurons with softmax activation to determine the probability of deception. All the fusion methods used in this study are an extension of this network. There are two reasons I use MLP as the base model of the proposed method. First, it is a simple and effective model widely used in various classification and regression tasks. Besides, MLP is the most basic deep learning model, and we can be easily combine it with other networks such as convolution neural network (CNN) or recurrent neural network (RNN). Therefore, we can use the proposed fusion method for complex input tasks like images (when combined with CNN) or sequential modeling tasks (combined with RNN)

In all the examples of this section, the modalities used for fusion are visual and acoustic. Denote the feature vectors for these modalities as x_v - visual, and x_a - acoustic. Early and late fusions of features are common ways to integrate multiple modalities. Network architectures of early and late fusion methods are shown in Figure 7. With the early fusion, we concatenate the vectors that contain acoustic and visual features into one single vector and then feed it to the MLP network. Early fusion is the most widely used combination method in previous studies for deception detection.

For the late fusion model, we can consider the network as a combination of two MLPs that is similar to early fusion; one network takes visual features as input, and the other network uses acoustic features. The outputs of these two subnetworks are then fully connected to a final output layer with two neurons. Models are trained separately among the two subnetworks and the network that connects them into the output; thus, results from two networks are considered in equal weight.

Hierarchical fusion is a method to combine different modalities proposed by [49] in a work for emotion recognition. Figure 8 describes the structure of this fusion method for the with an example of fusing visual and acoustic modalities.

As the name suggests, this fusion model's architecture resembles a hierarchy

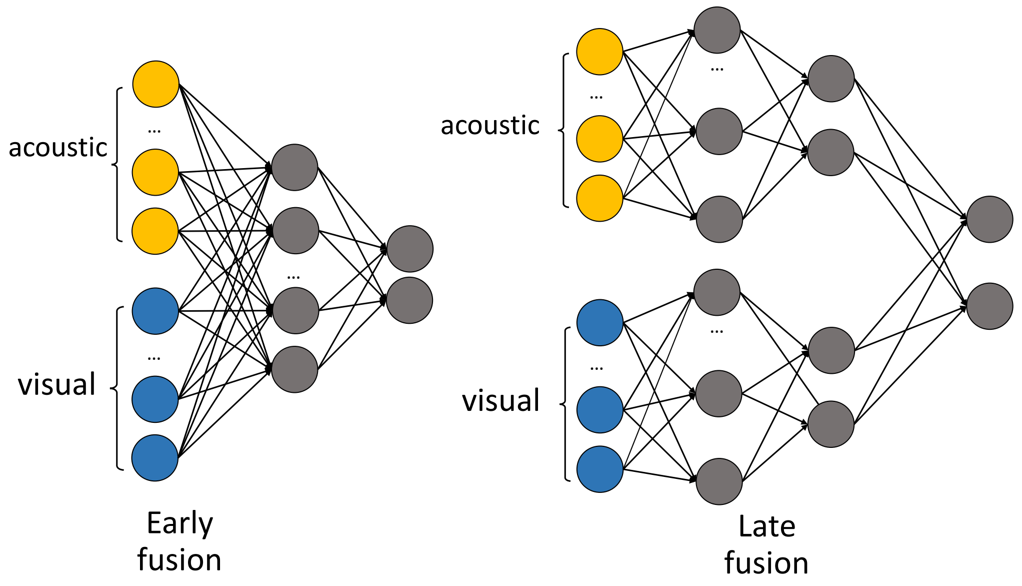

Figure 7. Early and late fusion network architecture.

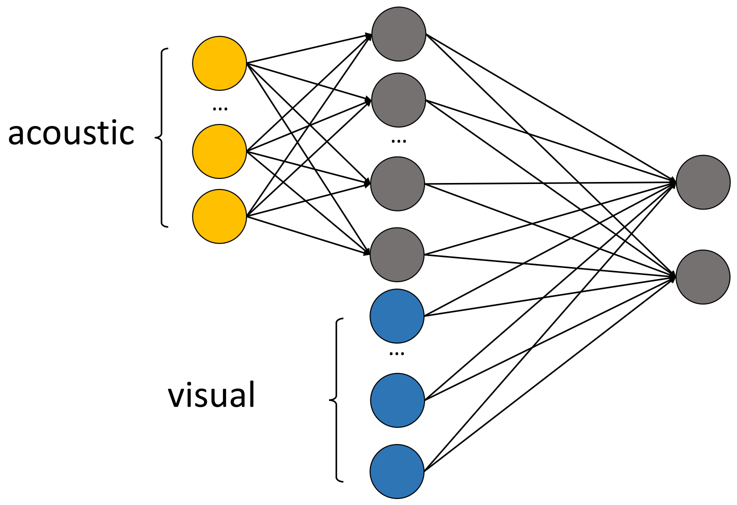

Figure 8. Hierarchical fusion architecture.

graph with the layers of network equivalent to levels in a hierarchy. In this example, the vector of acoustic features is fed into the input layer, which is fully connected to a hidden layer. The vector containing visual features is concatenated with this hidden layer, as shown in the figure. The resultant vector is fully connected to an output layer. Denote the subnetwork connects the acoustic input and hidden layer as f_{l1}, the subnetwork connects to output as f_{l2}, and the network's output as y, we have:

$$\begin{aligned} h_a &= f_{l1}(x_a) \\ y &= f_{l2}(h_a \oplus x_v) \end{aligned} \tag{8}$$

with $\oplus$ means concatenation of vectors. [49] argued that different modalities may describe data at different timescales or have different levels of abstraction, thus features from different modalities should be put into different layers of the network. In particular, the features that describe data at a larger timescale and are more abstract are used at higher levels, as shown in Figure 8. Detailed parameter settings are described in Section 3.3. I use similar numbers of parameters in different models to investigate the performances of model architectures.

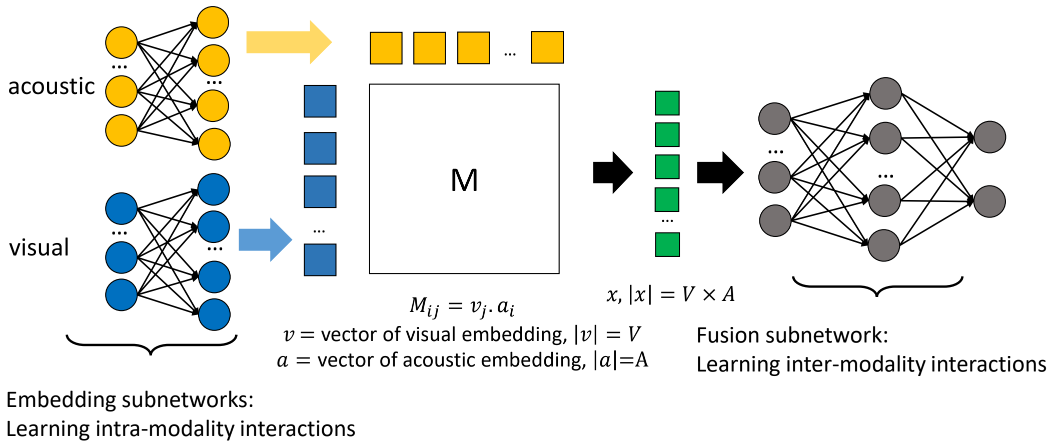
acoustic
visual
M
$M_{ij} = v_j . a_i$
v = vector of visual embedding, $|v| = V$
a = vector of acoustic embedding, $|a|$=A
$x, |x| = V \times A$
Embedding subnetworks:
Learning intra-modality interactions
Fusion subnetwork:
Learning inter-modality interactions

Figure 9. Tensor fusion architecture.

The tensor fusion method is another approach for combining different modalities in sentiment analysis [56] and face recognition [18]. [56] discussed that a neural network needs to learn about intra-modality and inter-modality feature interactions when combining multiple modalities. From Figure 7, we can see that the late fusion network cannot learn inter-modality interactions, because the features from different modalities are separated until the output layer. On the other hand, the early-fusion network learns both kinds of interactions simultaneously, so training is difficult. With TFN, learning of intra-modality and inter-modality interactions is separated, making the training process easier. Another benefit of TFN is that the representation of inter-modality interactions is given explicitly to the network in the form of the outer product (tensor), thus reducing training complexity. Because of these reasons, we expect TFN to work better than early and late fusion methods. Both studies [56, 18] observed that the tensor fusion outperformed early and late fusions in their respective tasks.

A TFN contains two kinds of subnetworks in its structure. The first one is the embedding subnetwork, which performs learning of intra-modality interactions. The outputs of those subnetworks are embedding vectors for each modality. Next, we perform outer production of the embedding vectors (visual and acoustic). The reason I use outer product to combine the vectors is that it can represent all the interactions between each feature from visual and acoustic modalities. The result is a matrix $M : V \times A$ (V and A are the size of the visual and acoustic embedding vectors, respectively), which is then flattened into a vector x with size $V \times A$. We feed this x into the fusion subnetwork (which is a MLP), which has the role of learning about the inter-modality interactions. Let us denote the visual, acoustic embedding subnetworks, and the fusion subnetwork as $f_{v_{emb}}, f_{a_{emb}}$ and f_{fusion}. The output y of TFN is given by:

$$\begin{aligned} v\_emb &= f_{v\_emb}(x_v) \\ a\_emb &= f_{a\_emb}(x_a) \\ h_{multi} &= v\_emb \otimes a\_emb \\ y &= f_{fusion}(h_{multi}) \end{aligned} \tag{9}$$

with $\otimes$ is outer product operation and h_{multi} is the resultant matrix M in Figure 9 after flattening. Previous studies reported that the decomposition of the tensor contributed to classification accuracies; however, I did not observe any improvement by the decomposition. Therefore, I only show the results of not using decomposition in the experiments.

3.2 Method

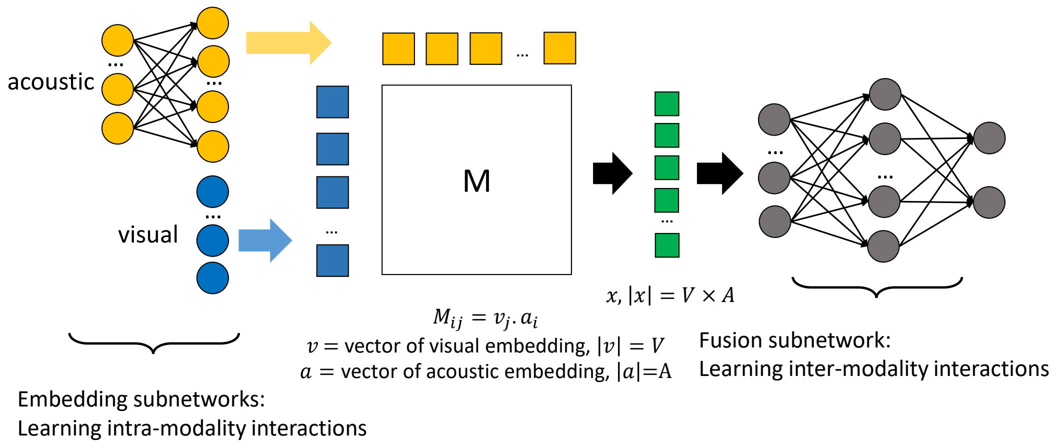
acoustic
visual
M
$x, |x| = V \times A$
$M_{ij} = v_j . a_i$
v = vector of visual embedding, $|v| = V$
a = vector of acoustic embedding, $|a|$=A
Fusion subnetwork:
Learning inter-modality interactions
Embedding subnetworks:
Learning intra-modality interactions

Figure 10. Hierarchical tensor fusion architecture.

The structure of the proposed network is shown in Figure 10. As can be seen from this figure, the hierarchical TFN structure is similar to a hierarchical network, but multi-modality fusion is performed by using the outer product (same as TFN) instead of concatenating. The proposed method's advantages over a hierarchical network is similar to that of TFN over early fusion thanks to the use of the outer product for fusion.

The hierarchical TFN also resembles a TFN structure. However, raw acoustic features are used as inputs of tensor fusion instead of an acoustic embedding vector, as shown in TFN. A substantial benefit over TFN that the proposed method has is reducing the complexity of network structure and the number of parameters. In some tasks (such as emotion recognition or deception detection), there can be a modality whose intra-modality interactions are not as beneficial as those of the other modalities. In such a situation, using a deep structured network to learn about these intra-modality interactions is superfluous. Therefore, by removing the embedding subnetwork for such modality, the whole system can focus on more important feature interactions and train a better model. In our case, empirical experiments show that visual feature interactions are less important than those of acoustic features. Thus, a vector containing raw visual features is used directly for fusion, as shown in Figure 10.

3.3 Evaluation I - Deception detection

In the first evaluation setting, I use the deception detection task to compare the efficiency of different multimodal fusion methods. Deception detection is a challenging task, even for humans, it is difficult to tell whether someone is lying or not without some prior information.

3.3.1 Dataset

The dataset I use for deception detection includes two types of data. The first one is taken from the Real-life Trial dataset [37]. I split the video into segments (as each segment contains a single utterance) to suit this research's purpose, which is to detect lies at utterance level. If the video was annotated with a lie label in the original dataset, I assign lie labels to all segments. I manually checked the segmented videos and found out that many have a low quality (blurry video or the speaker not facing the camera). Therefore, bad segments are filtered out by using the confidence score provided by the OpenFace toolkit [1]. In particular, if a video does not contain any frame with a confidence score of face tracking higher than 0.85, then it is automatically removed. After this process, from the trial dataset, we have 245 utterances; 105 of them are deceptive, and 140 of them are truthful. The second data came from a health consultation between two participants [32]. These data contain 844 truthful and 177 deceptive utterances. In total, the deception detection dataset used in the experiment includes 1265 utterances.

I perform this experiment of deception detection using 4-fold cross-validation. When checking the real-life trial dataset [36], I found out that many of the videos are taken from the same trial recording and were assigned the same label (lie or truth). It means there is a chance that the classifiers learn to predict recordings instead of deception labels if the partitioning of the dataset into train, development, and test sets are random. To avoid such a problem, I separate the samples in this dataset by the recording that they belong to, from 70 recordings of the original dataset to five portions. Details of data separation is shown in Table 1.

In previous studies [31, 36, 16], samples were chosen so that the ratio of lie/truth is balance (1:1). I followed the same configuration and allocated the samples in such a way that development and test set have ratio of lie/truth close

Table 1. Dataset partitions.

	# recordings	Recording ID	# lie	# honest
Partition 1	3	1,2,13	66	77
Partition 2	11	3,4,14,16,17,18, 19,24,26,27,35	59	76
Partition 3	13	10,23,25,28,29, 30,31,32,33,34, 36,37,38	62	79
Partition 4	4	9,15,17,23	60	77
Partition 5	39	the remaining	38	671

to 1:1. Particularly, for each split in this cross-validation setup, I pick out two partitions (1-2, 2-3, 3-4, and 4-1) to be used as development and test set; the remaining three partitions are used as training set. In partition 5, the number of honest samples exceeds the number of lie samples by a large margin, therefore, I did not use this partition for development and testing. I used over-sampling to achieve 1:1 ratio between lie and truth samples used for training.

3.3.2 Features extraction

I use the OpenFace toolkit [1] to extract facial features, which include 14 face action unit (AU) regressions and 6 AU classification values as well as head position, and head direction parameters for each frame by using this toolkit. These values were then normalized and discretized into five different levels of intensity to be used as features for deception detection. Acoustic features were extracted from audio files using the OpenSMILE toolkit [14]. I use the Interspeech 2009 (IS09) emotion challenge standard feature-set as the acoustic features [42]. These features were also used in previous studies of the deception detection task [31, 28]. In conclusion, for each segment sample in the dataset, I am able to extract 78 visual and 384 acoustic features.

Table 2. Deception detection performance of models based on different fusion methods. Precision, Recall, and F1-score are measured for the positive (Lie) label.

Model	# layers	# parameters	Accuracy	Precision	Recall	F1-score
single acoustic	4	162,900	53.78%	0.4747	0.5000	0.4870
single visual	4	164,352	49.28%	0.4095	0.3525	0.3879
multi early	4	169,872	53.42%	0.4603	0.3566	0.4018
multi late	4	164,972	54.68%	0.4794	0.3811	0.4247
multi hierarchical	4	168.192	53.78%	0.4733	0.4713	0.4723
multi TFN	5	163,528	50.36%	0.4216	0.3525	0.3839
multi hierarchical TFN	5	166,448	**58.63%**	**0.5304**	**0.5000**	**0.5148**

3.3.3 Results of deception detection

In the first experiment, single-visual, single-acoustic, and early fusion models all have two hidden layers, with the first layer have 256 units. The hierarchical fusion model contains two hidden layers. The first layer is fully-connected with the input layer from acoustic modality. This hidden layer has 256 units and is concatenated with the input visual features vector. The resultant concatenation fully connects with the second hidden layer, which is in turn fully-connected with the output. For TFN models, the embedding subnetworks have one hidden layer which has the same number of units as the output embedding vector (32 units). Similar to that from the original work [56], I use ReLU as the activation function for the embedding subnetwork. The fusion subnetwork of TFN has one hidden layer. I did not augment the embedding vectors with 1 since the empirical experiment results showed no improvement. In other words, the TFN model used in this experiment is equivalent to $TFN_{bimodal}$ of visual and acoustic modalities from the prior work [56]. With hierarchical TFN, the embedding subnetwork has one hidden layer (256 units), and the output has 32 units. Similar to TFN model, the fusion subnetwork of hierarchical TFN also contains one hidden layer.

All the models were trained using the Adam optimizer [21] with a softmax cross entropy loss function. I use the loss value attained in the development dataset to tune up the learning rate; the remaining hyper-parameters are the default setting of Chainer[1]. I train the deception detection models using mini-batch, with a batch size of 16. The learning rate decreased by 10% at every epoch. The loss on the development set was used to determine the point to stop; if we do not see improvement of the development loss for 100 epochs, the training was stopped. All the models are configured to have a similar number of parameters, ranging from 163,000 to 170,000.

Table 2 summarizes the parameters and performance of different models in the deception detection task. In this table, "single" refers to the models that use only one modality (visual or acoustic), and "multi" refers to the models that use both modalities. Early, late, hierarchical, and TFN indicate their integration methods. The numbers are averaged from 4-fold cross-validation results. Accuracy is measured for both labels (truthful and deceptive). Precision, recall, and F1-score are measured for the deceptive label.

From these results, it is clear that the proposed hierarchical TFN has the highest overall performance, outperforming hierarchical fusion ($p < 0.05$) and TFN ($p < 0.05$). In particular, we can see a considerable improvement in precision and F1-score compared to the other models.

We can see that the single visual model performs a bit worse than the single acoustic model ($p \approx 0.105$) in terms of accuracy, while the precision, recall, and F1-score are much lower. This difference can be explained by how we extracted features from raw data. Acoustic features are extracted from raw audio at 100 frames per second (fps), while visual features are extracted at only 30 frames per second (since all videos are recorded at 30 fps). Therefore, with the same spoken uttThus, we can expect that the visual modality's contribution to performance is less than the acoustic one. Another reason for the disparity in performance between visual and acoustic modality is the nature of recordings in the recorded deception dataset. Within every recording, the speaker knows that their statement will be assessed for honesty, and thus, the speaker tries to conceal their facial expressions as much as possible to avoid getting caught lying. Hence, detecting deception from visual clues (facial expressions) is not trivial.

The results reported here indicate another benefit brought by the fusion of modalities by a hierarchical structure. If information from one modality is less

[1]https://chainer.org/

useful for the given task compared with other modalities, hierarchical fusion can balance this difference and improve multimodal fusion. The reason for this phenomenon is that, since we input the less effective modality's features into the deeper layer of a neural network, there are fewer parameters used for learning that modality's intra-interactions. Therefore, our network can focus on other modalities' interactions, which is more critical for the task and increases performance. This reason also explains why the proposed hierarchical tensor fusion works well with the dataset. Since visual modality interactions are less important than that of acoustic, by removing the subnetwork that learns intra-modality interactions of visual features, the training of the network becomes more efficient.

3.3.4 Relationship between deception detection accuracies and numbers of parameters

In this experiment, I assess the effect of network depth on deception detection performance of hierarchical structures. I measured the accuracy when changing the number of hidden layers in hierarchical fusion and hierarchical TFN models. With hierarchical fusion, the number of hidden layers refers to the number of layers between the resultant concatenation and the output layer. For hierarchical TFN, the number of hidden layers refers to the hidden layers of the fusion subnetwork. The 1-layer models are taken directly from the previous experiment. With the 1-layer models as a base structure, I add new hidden layers (each has 256 units) to construct the 2-layer and 3-layer models.

Figure 11 illustrates the relations between the accuracy and layer numbers for both the hierarchical architecture and the hierarchical tensor fusion network architecture. We can see that there is no significant gain by increasing the number of hidden layers from one, two, or three. It indicates that a deep structure does not always contributes to the accuracy of deception detection, especially for a small dataset as I am using. However, it is difficult to prepare a large-scale dataset for deception detection. I leave further analysis in this aspect for future works.

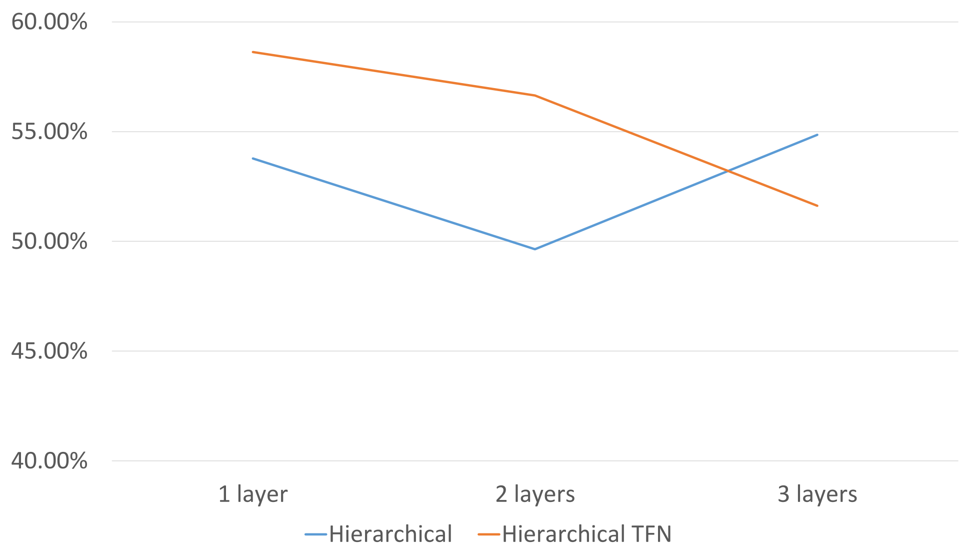

Figure 11. Effect of network depth on detection performance.

3.3.5 Dialog management performance based on predicted deception labels

In this experiment, I use predicted labels from deception detection models for a negotiation system that decides output dialog acts (DA) on the basis of the user's deception information (whether the user is lying or not). The dialog tactics of the system should change in accordance with the user deception; thus, I use a reinforcement learning based dialog manager that can change the system's dialog act by using the deceptive information of users [32]. This experiment was also conducted using the health consultation data from [32], where the system acts as the health consultant and persuades a human user to adopt a more healthy lifestyle.

I measured the system's performance by DA selection accuracy, which refers to the precision of the system's chosen dialog acts against reference actions chosen by a human when given the same user input utterance. All of the results shown in Table 3 used the policy trained with gold-labels of deception and the dialog act. Similar to the previous experiment, we can see that the negotiation system achieves the highest DA selection accuracy when using deception labels from

Table 3. Accuracy of dialog acts selection when using different deception labels result.

Deception labels used for dialog management	DA accuracy
chance rate deception	65.69%
gold-label deception	80.31%
single visual prediction	70.15%
single acoustic prediction	66.22%
multi early prediction	66.48%
multi late prediction	68.58%
multi hierarchical prediction	69.10%
multi TFN prediction	69.66%
multi hierarchical TFN prediction	71.20%

the proposed hierarchical TFN model. This result indicates that the proposed method contributes not only to deception detection but also helps the dialog system achieve high performance as well.

3.4 Evaluation II - Sentiment analysis

In this evaluation, I compare the hierarchical tensor fusion method's efficiency with current works in the sentiment analysis task. This task is a popular multimodal processing problem that has been used in existing studies.

3.4.1 Dataset

The dataset that I use for the experiment is the CMU Multimodal Opinion Sentiment and Emotion Intensity (CMU-MOSEI) [57]. This dataset consists of YouTube videos in which the speakers express their opinions about topics of interest. Each video contains only one single speaker who is facing the camera directly. This condition ensures that the speaker's expressions can be captured clearly for feature extraction. Fourteen expert judges manually check the quality of video, audio, and transcript of the acquired videos. After this manual quality inspection, a set of 3228 videos remained and were used as the dataset. The statistics of the CMU-MOSEI dataset are given in Table 3.4.1.

Table 4. Detail statistics of the CMU-MOSEI dataset.

Total number of videos	3228
Total number of sentences	23453
Total number of distinct speaker	1000
Total number of topics	250
Average number of sentence in a video	7.3
Average length of a sentence in seconds	7.28
Total number of words	447143
Total number of unique words	23026
Total number of words with frequency > 10	3413
Total number of words with frequency > 20	1971
Total number of words with frequency > 50	888

Each sentence is annotated with a sentiment value in the $[-3, 3]$ Likert scale of: [-3: highly negative, -2: negative, -1: weakly negative, 0: neutral, 1: weakly positive, 2: positive, 3: highly positive]. Three annotators annotated the sentiment values, and the average results are used as final sentiment values. Note that sentences from the same video can have different sentiment values. From this dataset, 16237 sentences are used for training set, 1871 are used for development set, and the remaining 4662 sentences are used for testing.

3.4.2 Feature extraction and training details

The CMU-MOSEI dataset were provided with pre-extracted features of visual, acoustic, and linguistic of each sentence. Details of feature extraction are as follows.

Visual. Frames are extracted from the full videos at 30Hz. The visual features follow the Facial Action Coding System (FACS) [12]. Additionally, a set of six basic emotions extracted from static faces using Emotient FACET. The OpenFace toolkit is used to extract facial landmarks, facial shape parameters, head pose, and eye gaze [1].

Acoustic. Acoustic features in terms of Mel-frequence coefficient, pitch, glottal source parameters, are extracted using the COVAREP software [10].

Linguistic. The provided manual transcriptions of each video are used for linguistic features. Glove word embeddings [35] were used to extract word vectors from transcripts. Features in terms of visual and acoustic are aligned based on the words in each sentence. As a result, each sentence has three sequences of visual, acoustic, and linguistic features with the same length.

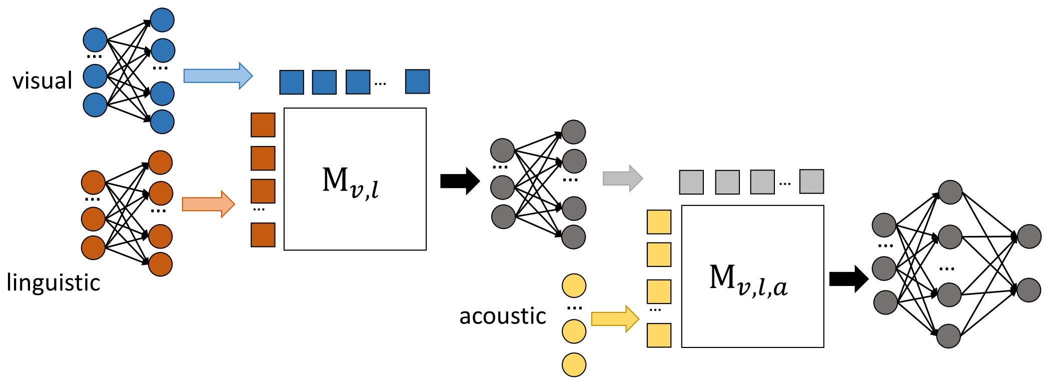

Figure 12. Hierarchical tensor fusion network for sentiment analysis task with three modalities.

The network model for the proposed hierarchical tensor fusion in this task is shown in Figure 12. First, visual and linguistic features are fused using hierarchical tensor fusion. Note that the embedding network for visual features has one hidden layer. On the other hand, the input and output layer of the embedding network for linguistic features are fully connected to each other. The resultant vector of visual and linguistic features - $M_{v,l}$ is fed into a fully-connected layer which outputs a visual-linguistic embedding. After that, we fuse this embedding vector with the acoustic features using outer product. Finally, the resultant fusion vector for all three modalities - $M_{v,l,a}$ is fed into an MLP with one hidden layer and output the sentiment regression value. For the hierarchical tensor fusion, visual, linguistic, and acoustic features are respectively fed into the first, second, and third hidden layer of the network. Fusion is performed by concatenation at

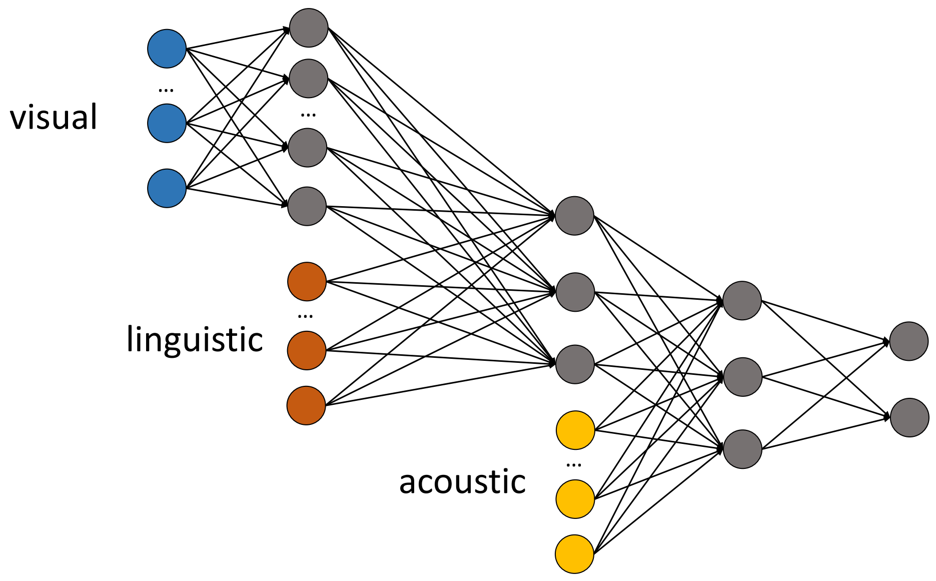

Figure 13. Hierarchical fusion network for sentiment analysis task with three modalities.

the third layer as shown in Figure 13. One hidden layer is connected to this fusion layer and the output.

With the tensor fusion network, the embedding network for visual features has one hidden layer. The embedding networks for acoustic and linguistic both have the output fully connected to the input layer. The fusion subnetwork is an MLP with one hidden layer.

I use Adam optimizer for training the parameters of all the models used in this experiment. The learning rate is set to 1e-3 at the beginning, then anneal toward 1e-4. The parameters are updated using gradients calculated from smooth L1-loss with $delta = 1$. Note that, similar to the deception detection evaluation, in this experiment, I set the size of hidden layers and embedding vectors so that all three models (Hierarchical TFN, Hierarchical, and TFN) have the same number of parameters, which is around 1.3 million.

Table 5. Results of sentiment analysis on the CMU-MOSEI dataset. $\uparrow$ indicates metrics that higher is better. $\downarrow$ indicates metrics that lower is better.

Model	$A^2 \uparrow$	F1-score $\uparrow$	$A^5 \uparrow$	$A^7 \uparrow$	MAE $\downarrow$	r $\uparrow$
Hierarchical [49]	76.19	82.58	46.53	45.49	0.67	0.61
Tensor fusion [56]	**79.94**	**85.84**	48.54	47.98	0.64	0.64
Hierarchical tensor fusion	79.08	85.11	**51.00**	**49.91**	**0.62**	**0.67**

3.4.3 Results

The results of this experiment are shown in Table 5. The performance is reported using six metrics. A^2, A^5, A^7 refers to accuracy of binary, 5-class, and 7-class classification results. For binary classification, if the sentiment values from both prediction and groundtruth are less than zero then the class is -1, otherwise the sentiment class is 1. In Table 5, I report both accuracy and F1-score of binary classification. In 7-class, the sentiment classes are created by rounding the sentiment values in the range [-3,3] into the nearest natural numbers. For 5-class, we first clip the sentiment values into the range [-2,2], then apply the same strategy used for making the labels in 7-class. MAE metric refers to mean absolute error. Finally, the Pearson's correlation r between the predictions and groudtruths is also reported. Note that since the samples in CMU-MOSEI dataset are sequential, a convolutional neural network with two layers is used to extract sentence-level features for visual, acoustic, and linguistic modalities. These features are used as input of the models in this experiment. As can be seen from Table 5, the proposed hierarchical tensor fusion achieves the highest performance in almost every metrics. Since the sentiment analysis task is modeled as a regression problem, A^5, A^7, MAE, and correlation are more accurate metrics to represent the performance of the models. Therefore, the hierarchical tensor fusion has the best performance overall in this task. These results show the generality of the proposed fusion method and confirm its efficiency in combining multimodal features compared to previous methods.

3.5 Conclusion

In this study about multimodal fusion, I described the hierarchical tensor fusion method, a novel approach for combining features from different modalities. The network in Figure 12 is not the only way to set up the network model configuration for the proposed hierarchical tensor fusion; there are various ways to configure the network's structure. For example, instead of using a layer to learn an embedding of visual and linguistic from $M_{v,l}$ and fuse all modalities by tensor product, we can directly perform a fusion of $M_{v,l}$ and the acoustic embedding with the tensor product operation. In addition, for different tasks, the layer where we input the same modality can be different. An example is shown in Figure 10, where the acoustic features are fed into the first layer while in Figure 12, the same features are input into the deeper layer. This difference is due to the abstraction level of the same modality that can change with different tasks, and thus the structure of the network should change accordingly. A simple way to find out the correct network structure is by trying out each configuration and compare the performance on a development dataset. Finally, the evaluations with deception detection and sentiment analysis tasks in 3.3 and 3.4 show that the proposed method is more efficient and outperforms previous works significantly.

4. Dialog management with multimodal information

In this chapter, I discuss the modeling for multimodal dialog management using modular-based and end-to-end approaches in the use-case of health consultation. The first section describes a modular-based dialog manager for the health consultation task, which was described in Chapter 2. In this chapter, I discuss the modeling for multimodal dialog management using modular-based and end-to-end approaches in the use-case of health consultation. Section 4.3, 4.4, and 4.5 describe details of user simulation, RL algorithm used for training the policy and performance metrics. Finally, a comparison of modular-based and end-to-end dialog managers' performance is provided in the evaluation section.

4.1 Modular-based approach

The structure of the modular-based dialog system is shown in Figure 14.Since the information of the user's deception is essential in the health consultation task, the multimodal processing that the system uses is a deception detector. This module's input is similar to the visual and acoustic features used in the deception detection experiment in the previous chapter. The deception detection module outputs a deception label, which is used by DST for dialog state tracking. This modular-based dialog system uses an explicit representation for the dialog state. In particular, the dialog state s is represented by a tuple (u, d), with u and d are the user's dialog action and user's deception respectively. u, d can be seen as the observations in this formulation of dialog management.

In order to find the best strategy for the dialog system against the user's deceptive behavior, it is necessary to consider errors in the prediction of the deception detection module. As shown in Chapter 3, all deception detection models do not have very high performance. The best model created from hierarchical tensor fusion still achieves less than 60% prediction accuracy. The partially observable Markov decision process (POMDP) is widely used to learn dialog systems' best strategy for such error-containing observations [55]. In POMDP, the action selection procedure is based on the belief state b_t - which is a probability distribution over all possible dialog states. In other words, we replace state s_t in the reinforce-

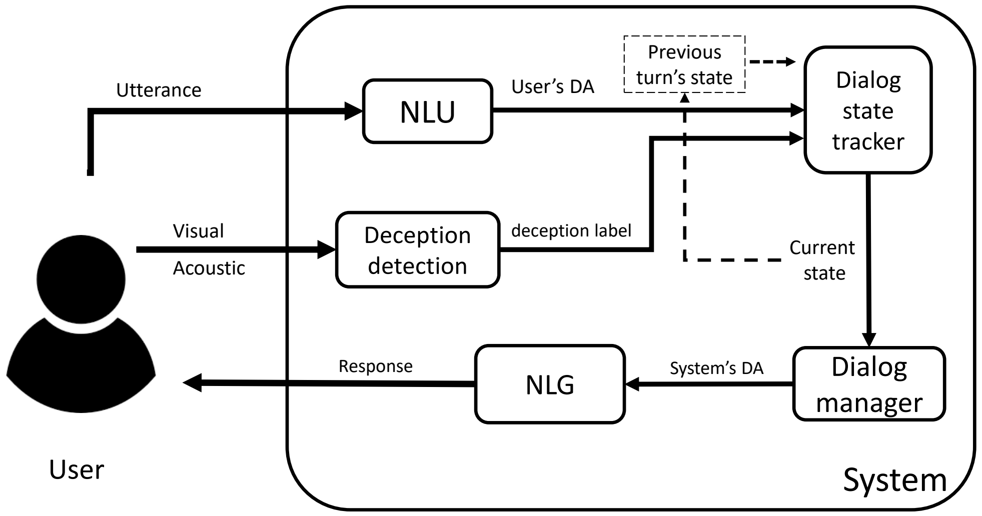

Figure 14. Modular dialog system for the health consultation task.

ment learning formulation described in Chapter 2 with the belief b_t. This belief is updated at every turn by the DST using the current observation u_t, d_t and the previous belief b_{t-1}. The belief state is represented by a 4-tuple $u, P(u), d, P(d)$ with u is the user's action with the highest probability in the user's action set and $d = 0$. In order to simplify the representation further, the probabilities $P(u), P(d)$ are rounded by 0.1. For example, if $P(u) = 0.45$ and $P(d) = 0.63$, the state representation will be $u, 0.5, d, 0.6$. This strategy of representing the belief state allows us to have a compact finite state space and still retain important information for action selection.

In this research, the dialog manager is trained using tabular Q-learning [50], a popular method to train the optimal policy π^*. For the proposed system, the training of Q-learning is done by iteratively performing the following update:

$$Q(b_t, a_t) \leftarrow (1 - \epsilon)Q(b_t, a_t) + \epsilon[R(b_t, a_t) + \gamma \max_{a_{t+1}} Q(b_{t+1}, a_{t+1})], \tag{10}$$

where ϵ is the learning rate, γ is the discount factor, and $R(b_t, a_t)$ is the reward the system receives when it performs an action a_t, given a dialog belief state b_t. While the deception detection modules are created using deep learning, I choose Q-learning with the Q-table to approximate the state-action value for two reasons.First, the state representation contains only two values, the user's dialog actions, and the user's deception. Thus, deep learning-based RL algorithms such as deep Q-network [30] is unnecessary. Furthermore, tabular Q-learning can actually learn the true optimal policy, provided that all state-action pairs are visited during the training process. On the other hand, deep Q-network uses gradient descent to optimize its parameters; thus, it can only guarantee local convergence. In summary, if we use the explicit state representation that is low-dimension, using a deep reinforcement learning solution is a bad choice.

The reward function is shown in Table 6. I design the system's rewards based on two points: the system follows the desired behavior (Figure 5). The system successfully persuades the user to accept its recommendation truly. A dialog is considered to be successful if the user honestly accepts to change his/her habits. This condition is equivalent to the user's dialog action is Accept, and the user's deception label is 0 (honest) in Table 6.

Table 6. Rewards in each turn for all possible combination of (s, a) in the state-action space.

Dialog state		Rewards		
User DA (s)	d	Offer	Framing	End
Accept	0	–10	–10	+100
	1	–10	+10	-100
Reject	0	+10	+10	–100
	1	–10	+10	–100
Question	0	–10	+10	–100
Hesitate	0	+10	+10	–100

4.2 End-to-end approach

Before going into details of the end-to-end dialog management approach, I use the health consultation task as a motivation example of how this approach is more beneficial than the modular-based approach.

Let us consider the situation where we have already built a health consultation dialog system using the modular-based approach. After analyzing the system's performance, we decide to incorporate the user's emotion information into the dialog management process. Such changes require the following steps:

1. Construction of an emotion recognition module.

2. Re-annotate the dialog dataset with emotion labels and build a new user simulator.

3. Train the DST and policy manager modules.

The problem lies in the first step, where we need to construct the emotion recognition module.Like deception detection, this task is also difficult, and building a good emotion detector is time-consuming. When we train the end-to-end dialog manager, information on the user's emotion will be learned automatically by interacting with the new user simulator. The manager can incorporate emotion information into the latent dialog state without explicit emotion labels. Therefore, the end-to-end approach does not require construction of an emotion recognition module, saving us much time for the system overhaul.

Figure 15 shows the structure of the end-to-end dialog manager for the task of health consultation. The input observations are denoted by o^l, o^v, o^a, which denotes the observation for linguistic, visual, and acoustic modality, respectively.

Here in this end-to-end model, the hierarchical tensor fusion is directly integrated into the end-to-end dialog manager. In practice, this integration is performed by connect a hierarchical tensor network with the input of LSTM layer. This fusion network is represented by the component *Hier-TFN* in Figure 15. The role of this component is to summarize input multimodal observation features into a high-level representation, the *multimodal observation*. Therefore, with the multimodal end-to-end dialog manager, input of state tracking is this "latent" multimodal observation instead of the raw observation features, as shown in Figure 4. Next, the LSTM layer performs state tracking from this multimodal observation and outputs the dialog state. Finally, the policy select the most appropriate system's action a_t as the response. The neural network that represents the end-to-end dialog manager in Figure 15 can be seen as a combination of three

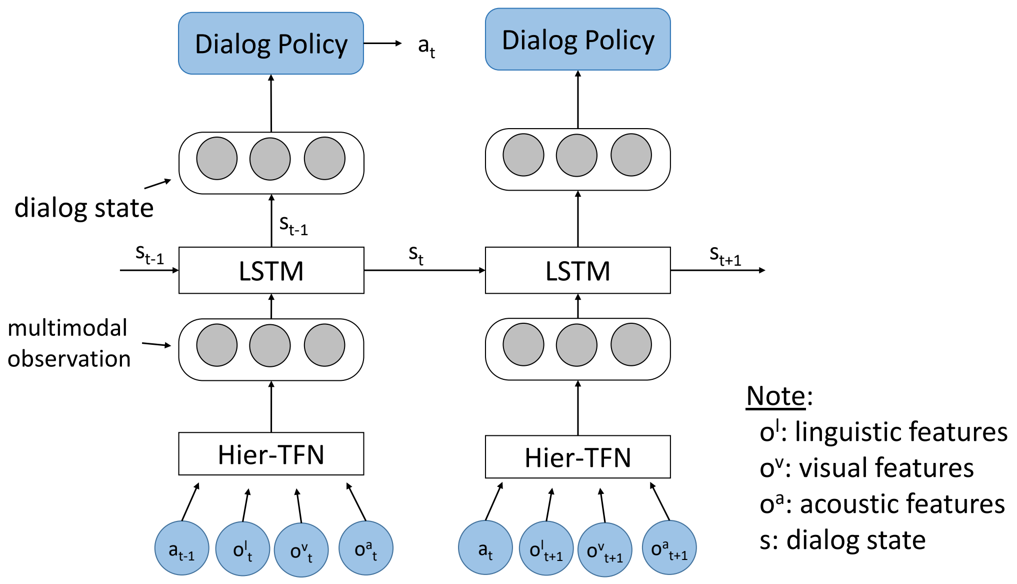

Figure 15. Multimodal dialog management using the end-to-end approach.

"components": the multimodal fusion (Hier-TFN), the state tracker(LSTM), and the dialog policy.

4.3 Multimodal user simulation

This section explain the construction of user simulators for training the policies for the task of health consultation. In this task, the user can use deceptions in the negotiation with the system; thus, I create a user simulator that generates labels of user's action u and deception information d with the following intention and deception models:

$$\begin{aligned} \text{intention model} &= P(u_{t+1}|u_t, d_t, d_{t+1}, a_t) \\ \text{deception model} &= P(d_{t+1}|u_t, d_t, a_t) \end{aligned} \tag{11}$$

The probability distributions in Equation 11 are estimated by using maximum likelihood from the health consultation dialog corpus. I provide more details about this dataset in Section 4.5. Figure 14 shows the full system structure that can interact with the human user naturally. However, in the training of dialog manager modules, natural language understanding and deception detection are not necessary. Instead, the dialog state tracker takes the input of the user's action u and user's deception d directly from the user simulator. To recreate the uncertainty in prediction of deception detection module, the probabilities $P(u)$ and $P(s)$ are randomized from the range $[0, 1]$.

For the end-to-end dialog manager's training, I use the same user simulator that was defined in Equation 11. Let us recall that in each turn, the dialog manager takes input features (observations) from three modalities: linguistic, visual, and acoustic. As a result, It is not possible to directly use outputs of user simulator for training the policy in the case of end-to-end dialog management. Zhao and Eskenazie (2016) use random sampling of user utterances from dialog corpus given the user's action generated by the simulator. In this research, I adopt the same strategy but with some modification for multimodal dialog. Particularly, I use the user's action u as the linguistic observation o^l directly. Given the pair of u and d, the set of tuples o^v, o^a for this specific u, d are extracted from the dialog corpus. After that, one tuple is randomly selected as input for training the policy. More details about the visual and acoustic observations are given in Section 4.5.

The simulation modeling described above is not the only way to build a user simulator. There are many other choices, such as using supervised generative models. In addition to that, it is not compulsory to have high-level multimodal labels such as deception or emotion to build the user simulator. We can build a user simulator with generative modeling that directly generates raw features (observations) without generating multimodal labels. The user simulation modeling method described in this section is the most popular and simple [41, 55, 34]; thus, I used it for my research. In general, the more accurate the user simulator imitates the real human user's behavior, the better policy we can train. The construction of a multimodal user simulator is also an interesting topic; however, I leave further investigation for future works.

4.4 Dialog management training with policy gradient methods

In principle, any reinforcement learning algorithm can be used for training the dialog policy. In this research, I use the REINFORCE algorithm [53] for policy learning. This algorithm is a classic solution that has been used for various problems, not just in reinforcement learning. Even though many other algorithms have been developed since the REINFORCE proposal, this algorithm remains a popular choice in reinforcement learning research due to its simplicity and good performance.

The REINFORCE algorithm belongs to the *policy gradient* class of reinforcement learning algorithms [46]. The training of the policy in REINFORCE, or other policy gradient-based algorithms, is called *on-policy learning*. This is because the interaction samples used for the training is from the same policy that we are optimizing. Compared to off-policy methods such as Q-learning or DQN [50, 30], on-policy algorithms like REINFORCE are simpler in general, have less variance, and are faster to converge. These are the reasons why I choose REINFORCE as the training algorithm in this research.

Let us denote the parameters of the policy as θ. With REINFORCE algorithm, we update θ with the following gradient,

$$\begin{aligned}\partial(\theta) &= E\left[\sum_{t=0}^{T} R_t \frac{\partial \pi^\theta(s_t, a_t)}{\pi^\theta(s_t, a_t)\partial\theta}\middle| \pi\right] \\ &= E\left[\sum_{t=0}^{T} R_t \frac{\partial log \pi^\theta(s_t, a_t)}{\partial\theta}\middle| \pi\right]\end{aligned} \tag{12}$$

with R_t is the return at time step t, as defined in Chapter 2 and $\pi^\theta(s, a)$ denotes policy with its parameters θ. In general, updating the parameters using 12 has large variance, especially with one-sample update (the policy is updated after each episode). Thus, a variance reduction method was introduced [54, 25] by using a *baseline* $b(s_t)$, and the gradient becomes,

$$\partial(\theta) = E\left[\sum_{t=0}^{T}(R_t - b(s_t)) \frac{\partial log \pi^\theta(s_t, a_t)}{\partial\theta}\middle| \pi\right] \tag{13}$$

The baseline is not necessary a function of s_t, it can be a random variable or even a constant number, as long as it does not vary with a. In the appendix, I show why including the baseline does not affect estimation of the gradient $\partial(\theta)$. [46] show that the most natural and effective baseline is actually the state-value function $V^{\pi}(s)$. Replace this into Equation 13, we have the true gradient update, which is given as,

$$\partial(\theta) = E\left[\sum_{t=0}^{T}(R_t - V^{\pi}(s_t))\frac{\partial log\pi^{\theta}(s_t, a_t)}{\partial\theta}\middle|\pi\right] \tag{14}$$

4.5 Evaluation

4.5.1 Dataset

I collected a dialog dataset for the health consultation scenario using two settings: Wizard-of-Oz and direct conversation. The details for each setting are provided below. The corpus that I collected consists of dialogs between two students fluent in English. Both participants role-played as consultant and consultee, and the consultant tried to persuade the consultee to change some actual living habits. All participants were working in the same academic environment at the time of data collection. A total of seven participants took part in the recordings. Four of them played the role of consultant (system), and six played the role of consultee (user).

Each recording session was carried out by two participants, one playing the system's role and the other of the user. Each session consisted of six dialogs for each of the living habit topics. The participants who played the consultee's role were given payment as a reward for the outcome of the conversation. If they pretended to agree with the system's offer, they would receive a lower payment; otherwise, the participant would get a higher payment with the condition that they would need to adopt the new habit for at least one week. The payment was intended to create a situation where the user has to choose between an easy activity (continuing a current habit) with low reward and challenging activity with a higher reward (changing to a new habit) to observe more lies.

In the WoZ setting, the two participants sit in front of a laptop in two separate rooms and cannot see each other. The utterances spoken by a participant in one room were transmitted to the other room and output by a speaker. The recorded WoZ data's total duration is about three hours and 20 minutes long and contained 35 dialogs, with an average of 5.8 turns per dialog. One expert annotated DA labels, and deception labels were provided by the participant who made the deceptions.

The recording setup was similar to the WoZ scenario with the direct conversation setting but without TTS, since the participants were now talking directly with each other. The dialogs collected using this setup are more complex and more helpful for assessing dialog management performance. The direct conversation dataset's total duration is about two hours and 35 minutes in length and contained 36 dialogs, with an average of 4.94 turns per dialog. 7 shows the statistics for both type of recorded data. For the experiments, I use the WoZ data to create the user simulator used for training the policies. On the other hand, the direct conversation data is used to create the simulator for evaluation.

Table 7. Statistics of deception labels and dialog acts in the health consultation corpus.

Data		Train	Test
Consultant DA	End	14.43%	17.54%
	Framing	43.30%	36.26%
	Offer	42.27%	46.20%
Consultee DA	Hesitate	21.69%	17.64%
	Question	3.61%	9.86%
	Accept	51.81%	19.72%
	Reject	22.89%	52.82%
% lie in user utterances		18.07%	19.72%

4.5.2 Observations/Features extraction

The observations for end-to-end dialog managers are created as follows. Let us recall that in the health consultation dataset, . In each turn, I split the recorded video of the user into 30 segments, and sample one frame randomly from each segment. I use the OpenFace toolkit [1] to extract 14 face action unit (AU) regressions and 6 AU classification values for each frame. The visual observation at each turn o^v is the vector that contains these extracted values. Acoustic features are extracted from the audio using the OpenSMILE toolkit [14] with the Interspeech 2009 (IS09) emotion challenge standard feature-set as the feature template [42]. I use these extracted features to create the acoustic observation o^a. As explain above in Section 4.3, observations o_v and o_a are sampled uniformly from the dataset with the condition of u and d, which are generated by the user simulator.

4.5.3 Results

Even though the input dialog managers based on modular and end-to-end approaches are different since I use the same distribution models (intention and deception model); the evaluation condition is fair.

In this experiment, I use the dialog action selection accuracy as the measurement for dialog management performance. The results are shown in Table 8. As can be seen, there are no significant differences between the modular-based and end-to-end managers' performance. This result means that the end-to-end dialog manager successfully incorporates the user's deceptive behavior in its latent state representation without using deception labels as in the modular-based manager.

Table 8. Accuracy of dialog act selection when using different approaches.

Dialog manager models	DA selection accuracy
Modular-based	71.20%
End-to-end	69.95%

5. Policy reuse with action relation probability

Current policy adaptation studies in reinforcement learning-based dialog management follow the "weight initialization" approach [19, 6, 29]. As explained in Chapter 1, this strategy requires us to train the target policy from scratch with an RL algorithm, which is a tedious process and needs lots of effort, especially for complex tasks. Besides, let us recall that the construction of dialog policy usually involves user simulation. When we have only a small amount of data in the target dataset, the user simulator does not well-represent the real human user's behavior. Therefore, the policy that we train can have high performance against the simulator but does not work well versus real human users.

Problem statement. Given a source task with the state space S_A and the action set A. Let us assume that we have trained a policy $\pi(s, a)$ for the source task. The target task's state space and action set are denoted as S_B and B, respectively. Is it possible to derive a policy $\pi(s, b)$ for the target task from $\pi(s, a)$, without RL training?

This chapter provides an answer to this question, which is the dialog policy reuse algorithm, a novel adaptation method that does not rely on RL training.

5.1 Related works

There have been a variety of studies about policy adaptation for reinforcement learning-based goal-oriented dialog management.

[6] proposes a policy adaptation method by using a multi-agent dialog policy. [6] uses an explicit representation of the dialog state, which contains multiple slot information. For each slot, we build an "agent" that learns how to choose actions corresponding to this slot. The dialog policy is an ensemble of these agents. In the target task, for each new slot information, the network weights of its corresponding agent are initialized using weights of the agents that have been trained in the source task. While this adaptation method does not require the state representation to be similar, as in DPRA, it has the restriction that state representation of both tasks must contain values of slot information. Also, this method requires the action set for each agent to be the same in order to perform weight initialization. This requirement makes [6] not as flexible as DPRA in

terms of action space restriction.

[19] introduces a new policy adaptation method by using weight bootstrapping. [19] also uses slot information for the state representation. This method proposes sharing the slots and actions across the source, and the target task A neural network is constructed with the input layer has the number of neurons equal to the number of total grouped slots. Similarly, the output layer has the number of neurons equal to the number of total grouped actions. First, we train a policy on source task by reinforcement learning. The network weights are then fine-tuned by training on the target task. This adaptation method has the same restriction of state representation as in [6]. Furthermore, [19] also requires overlapping the source task's action sets and the target task. The proposed method, DPRA, is more flexible and does not have this limitation.

[29] proposes an adaptation of dialog policy by using action embeddings. They argue that a set of action embeddings can be shared across the source and the target tasks. In practice, action embedding is represented by a hidden layer that is fully connected to both the input and output layers. After training the policies from all the source tasks, the weight parameters that connect the input layer and the hidden layer are used to initialize the corresponding weights in the target policy's neural network. I use this method as the baseline because this adaptation method does not require any additional knowledge or relations between the source and the target task and is comparable to the proposed method in terms of flexibility.

5.2 Method

This section answers the problem statement that was raised previously. I particularly show that it is possible to learn a policy for the target task without RL. Instead of training policy by interactions with a user simulator, we can establish a connection between the source task's policies and the target task through a special mapping distribution, which is called *action-relation probability.* The proposed adaptation method, DPRA, learns this distribution from the dialog samples in both tasks and can immediately derive a policy for the target task. Thus, DPRA can reduce learning time remarkably. Since the proposed method does not use the user simulator, we can avoid a low performance due to errors in constructing the user simulator.

5.2.1 Policy adaptation with action-relation probability

We consider the policy adaptation from a source task to a target task. First, we make the following assumptions, which allow derivation of relation between the source's and the target task's policies:

Assumption 1: The source task and the target task share the same state space S.

This assumption is not restrictive. The union space $S = S_A \cup S_B$ is the state space that satisfies the assumption.

Assumption 2: The source task and the target task has similar state representation.

As defined above, the *observation* refers to the information necessary for the agent's action selection in each time step. An example of the observation is the features extracted from the user's utterance at each turn. Obviously, in policy adaptation settings, the source and target task are different and require a distinctive set of features. Thus, the source task's state representation is also not the same as the target task's. However, we can define a unified set of features for those tasks; therefore, we can have the same observation across the source and the target tasks.

We can establish a connection between the source and the target task's policy with the following equation:

Let us denote the source task's policy as $\pi(s, a)$ and the target task's policy as $\pi(s, b)$, with $a \in A, b \in B$ are action set in the source and target task respectively. We have,

$$\pi(s, b) = \sum_{a \in A} P(b|a, s)\pi(s, a). \tag{15}$$

Equation 15 says that with any conditional distribution of $P(b|a, s)$, from the source policy $\pi(s, a)$, we can directly infer a policy for the target task $\pi(s, b)$ without RL training. This distribution $P(b|a, s)$ is termed *action-relation probability.* The proposed method of policy adaptation models this distribution instead of performing RL training.

5.2.2 Action-relation probability modeling with mixture density network

This section explains a modeling of action-relation probability distribution. First, let us denote the state and action at time step t as $s = s_t$ and $a = a_t$ respectively, the state at the next time step is $s' = s_{t+1}$. We have the state transition in the target task is given as,

$$\begin{aligned} P(s'|a,s) &= \sum_{b\in B} P(s',b|a,s) \quad (\textit{law of total probability}) \\ &= \sum_{b\in B} P(s'|b,a,s)P(b|a,s). \end{aligned} \tag{16}$$

The state transition of the source task has the form of a mixture model with the action-relation probability as the component weights. Mixture density network (MDN) [5], or MDN, is a suitable approach for modeling the state transition $P(s'|a,s)$. In principle, mixture density network is a type of Gaussian mixture model (GMM) that utilizes artificial neural network. Given multivariate random variables x, y, MDN models the conditional probability density $p(y|x)$ as,

$$p(y|x) = \sum_{m=1}^{M} w_m(x) \cdot \mathcal{N}(y; \mu_m(x), \sigma_m^2(x)). \tag{17}$$

M is the number of components, $w_m(x)$, $\mu_m(x)$, and $\sigma_m^2(x)$ are the component weight, mean, and standard deviation for component m. I make an assumption that that these mixture variables are functions of the input x and they are approximated by neural networks $f_m^w, f_m^\mu, f_m^\sigma$, parameterized by $\theta_m^w, \theta_m^\mu, \theta_m^\sigma$ respectively. With some slight abuse of notation, we have:

$$w_m(x) \approx \frac{\exp(\mathrm{f_m^w(x;\theta_m^w)})}{\sum_{l=1}^{M}\exp(\mathrm{f_l^w(x;\theta_l^w)})} \tag{18a}$$

$$\mu_m(x) \approx f_m^\mu(x;\theta_m^\mu) \tag{18b}$$

$$\sigma(x) \approx \exp(\mathrm{f_m^\sigma(x;\theta_m^\sigma)}) \tag{18c}$$

With MDN, we assume that all components in the multivariate random variable y are mutually independent, thus the covariance matrix is diagonal and can be represented by a vector with the same dimension as $f_m^\mu(x)$. Let us denote the dataset as $D = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}, i = 1..N$, with $\mathbf{x}, \mathbf{y}$ are observed data for random variables x and y. The parameters are optimized using gradient descent with the following negative log-likelihood:

$$L = -log(\prod_{i=1}^{N} p(\sum_{m=1}^{M} w_m(\mathbf{x^{(i)}}) \cdot \mathcal{N}(\mathbf{y}^{(i)}; \mu_m(\mathbf{x^{(i)}}), \sigma_m^2(\mathbf{x^{(i)}})))) \tag{19}$$

By replacing the probabilities with probability density functions, the mixture model in (16) is now given by,

$$\begin{aligned} p(s'|a_i, s) &= p_i(s'|s) \qquad (a_i \in A) \\ &= \sum_{j=1}^{|B|} P_{ij}(s) \cdot p_{ij}(s'|s) \\ &= \sum_{j=1}^{|B|} w_{ij}(s) \cdot \mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s)). \end{aligned} \tag{20}$$

An illustration of these mixture models are given in Figure 16. In principle, the density of state transition caused by a_i, is a mixture model with each component $p_{ij}(s'|s) = p(s'|a_i, b_j, s)$ follows a Gaussian distribution $\mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s))$. For each action a_i in the source task, we can train its corresponding MDN with just the samples (s', a_i, s). The action-relation probability $P(b = b_j|a = a_i, s)$ is approximated by $f^w(x; \theta_{ij}^w))$ as shown in (18a). However, in that case, we cannot guarantee that $\mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s))$ truly models the state transition $p(s'|a_i, b_j, s)$ since the source task samples do not contain information of b_j. A natural solution is to additionally train the components by using the samples (s', b_j, s) in the target task. I call this process "*component matching*" since it actually "matches" the distribution of Gaussian component $\mathcal{N}(s'; \mu_{ij}(s), \sigma_{ij}^2(s))$ to the transition of $p(s'|b_j, s)$.

In this work, I propose two methods of component matching. In the first method, I assume that $p(s'|a_i, b_j, s) = p(s'|b_j, s) \forall a_i \in A, b_j \in B$. With this assumption, we can perform component matching by simply optimizing the networks' parameters using negative log-likelihood of the target task's samples:

$$L = -log(\prod_{i=1}^{N} \mathcal{N}(\mathbf{s'}^{(i)}; \mu(\mathbf{s^{(i)}}), \sigma^2(\mathbf{s^{(i)}}))). \tag{21}$$

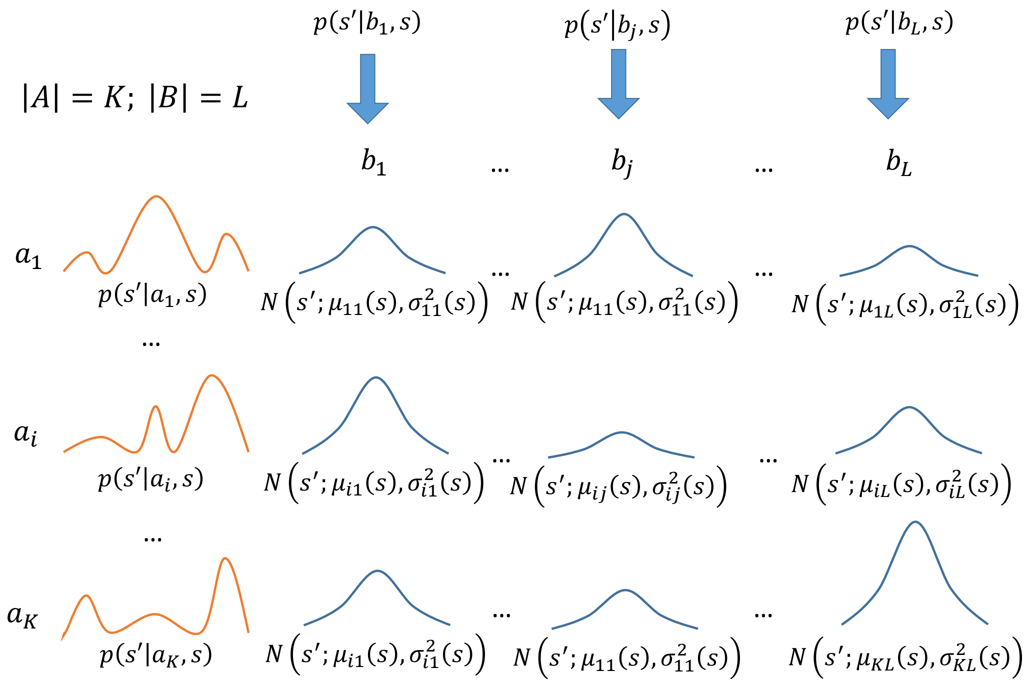
$p(s'|b_1, s)$
$p(s'|b_j, s)$
$p(s'|b_L, s)$
$|A| = K;\ |B| = L$
b_1
...
b_j
...
b_L
a_1
$p(s'|a_1, s)$
$N\left(s'; \mu_{11}(s), \sigma_{11}^2(s)\right)$
$N\left(s'; \mu_{11}(s), \sigma_{11}^2(s)\right)$
$N\left(s'; \mu_{1L}(s), \sigma_{1L}^2(s)\right)$
a_i
$p(s'|a_i, s)$
$N\left(s'; \mu_{i1}(s), \sigma_{i1}^2(s)\right)$
$N\left(s'; \mu_{ij}(s), \sigma_{ij}^2(s)\right)$
$N\left(s'; \mu_{iL}(s), \sigma_{iL}^2(s)\right)$
a_K
$p(s'|a_K, s)$
$N\left(s'; \mu_{i1}(s), \sigma_{i1}^2(s)\right)$
$N\left(s'; \mu_{11}(s), \sigma_{11}^2(s)\right)$
$N\left(s'; \mu_{KL}(s), \sigma_{KL}^2(s)\right)$

Figure 16. State transition modeling by mixture density network..

Since the training of this component matching method is similar to the training process of a regression model, I define it with the term *component matching by regression*. Algorithm 1 shows the pseudo code for the training process of the mixture model in Equation 20 by using component matching by regression.

Algorithm 1 Action-relation probability modeling by MDN with component matching by regression

```
Initialize the network weights $\theta^w, \theta^\mu, \theta^\sigma$ randomly
Initialize the MDN gradient $d\theta_{MDN} \leftarrow 0$
Initialize the gradient for component matching $d\theta_{CM} \leftarrow 0$
Initialize the iteration counter $t \leftarrow 0$
repeat
  for all $a_i$ such that $a_i \in A$ do
    for all $b_j$ such that $b_j \in B$ do
      Calculate the gradient $d\theta_{CM}$ by (21)
      Update parameters $\theta^\mu, \theta^\sigma$ with $d\theta_{CM}$
    end for
    Calculate the gradient $d\theta_{MDN}$ by (19)
    Update parameters $\theta^w, \theta^\mu, \theta^\sigma$ with $d\theta_{MDN}$
  end for
  $t \leftarrow t + 1$
until $t > t_{max}$
```

The second component matching method stems from the following derivation:

$$\begin{aligned} P(s'|b,s) &= \sum_{a \in A} P(s',a|b,s) \quad (\textit{law of total probability}) \\ &= \sum_{a \in A} P(s'|b,a,s)P(a|b,s). \end{aligned} \tag{22}$$

Equation (22) says that the transition distribution $P(s'|b,s)$ also has the form of a mixture model and we can model it by MDN. We can see that the mixture in Equation (22) has the same component $P(s'|a,b,s)$ as in (16), but different component weights $P(a|b,s)$. I define the parameter of the network that approximates the component weights as $\hat{\theta}^w$.

Algorithm 2 Action-relation probability modeling by MDN with component matching by MDN

```
Initialize the network weights $\theta^w, \hat{\theta}^w, \theta^\mu, \theta^\sigma$ randomly
Initialize the MDN gradient $d\theta_{MDN} \leftarrow 0$
Initialize the gradient for component matching $d\theta_{CM} \leftarrow 0$
Initialize the iteration counter $t \leftarrow 0$
repeat
  for all $a_i$ such that $a_i \in A$ do
    for all $b_j$ such that $b_j \in B$ do
      Calculate the gradient $d\theta_{CM}$ by (23)
      Update parameters $\hat{\theta}^w, \theta^\mu, \theta^\sigma$ with $d\theta_{CM}$
    end for
    Calculate the gradient $d\theta_{MDN}$ by (19)
    Update parameters $\theta^w, \theta^\mu, \theta^\sigma$ with $d\theta_{MDN}$
  end for
  $t \leftarrow t + 1$
until $t > t_{max}$
```

In principle, the components in column of b_j in Figure 16 form a mixture model for the density of state transition $p(s'|b_j, s)$. Similarly, we can train this mixture with a loss function that is similar to (19) by using the target task samples (s', b_j, s), which is given by,

$$L = -log(\prod_{i=1}^{N} p(\sum_{m=1}^{M} \hat{w}_m(\mathbf{x^{(i)}}) \cdot \mathcal{N}(\mathbf{y}^{(i)}; \mu_m(\mathbf{x^{(i)}}), \sigma_m^2(\mathbf{x^{(i)}})))). \tag{23}$$

Therefore, in this research, the term to describe this method is *component matching by MDN*. The pseudo-code for action-relation probability modeling using MDN component matching is shown in Algorithm 2.

Finally, the procedure of my proposed method, the dialog policy reuse algorithm, is shown in Algorithm 3:

```
Algorithm 3 Dialog policy reuse algorithm - DPRA
  Step 1: Train a policy $\pi(s, a)$ for source task
  Step 2: Model the action-relation probability $P(b|s, a)$ by using either Algorithm 1 or Algorithm 2
  Step 3: Create a policy for target task $\pi(s, b)$ , by using Theorem 15, with the action-relation probability learned in Step 2.
```

To summarize this section, I show that the resultant policy $\pi(s, b)$ of DPRA is a proper policy, which means:

$$\sum_{b \in B} \pi(s, b) = 1 \tag{24}$$

The proof for this statement is provided in the Appendix. Intuitively, DPRA works as follows. Given policy $\pi(s, a)$ in the source task, let us assume that the agent with *pi* selects action a given current state s. DPRA finds action b in the target task that causes similar transition $s \rightarrow s'$ of an action a in the source task, in other words $P(s'|s, a) \simeq P(s'|s, b)$. Instead of making a deterministic mapping, we learn a distribution that connects a to all available actions in the target task, which is $P(b|a, s)$. This is the reason why I use the term action-relation probability to define this special distribution. With the condition that the source and the target task are also similar in the rewards dynamic, if the learned policy $\pi(s, a)$ in the source task is optimal, then the policy $\pi(s, b)$ learned by DPRA is nearly optimal as well.

DPRA requires identical state space and state representation of the source and target tasks (Assumption 1 and 2). Recall that all policy adaptation methods

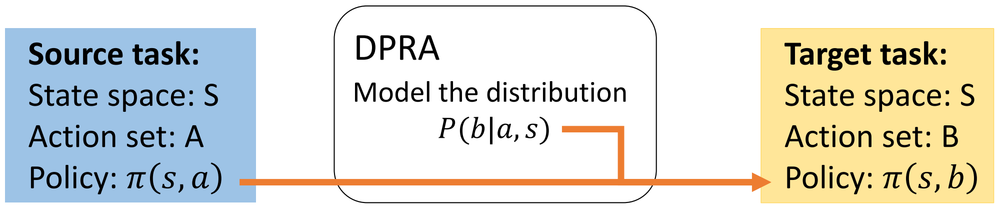

Figure 17. Working procedure of DPRA.

only work in cases where the source and target tasks are similar. In this situation, even if the two tasks' state spaces and state representations are not completely identical, we expect that they still share considerable similarities. Thus, the proposed method can still learn a good policy for the target task without the ideal conditions in Assumption 1 and 2.

In the context of a multimodal dialog, similarities in terms of modalities between the source and the target task are not always guaranteed. From the view of multi-modalities, the task differences come from the modality type difference or feature difference. However, it is reasonable to assume that if a modality is used in both the source and target task, the features extracted for this modality are similar in those tasks. Therefore, the differences in terms of modalities between the source and the target task only come from their modality types. Let us denote the set of modalities in the source task as M^S and the target task as M^T. There are three situations where M^S and M^T are different.

1. There exists $M_i \in M^S$ and $M_j \in M^T$ such that $M_i \notin M^T$ and $M_J \notin M^S$.
2. $M^T \subset M^S$ or $\forall M_j \in M^T, M_j \in M^S$.
3. $M^S \subset M^T$ or $\forall M_i \in M^S, M_i \in M^T$.

In all of the situations above, the condition of a similar state representation is not satisfied. However, we can alleviate this issue by using a "padding" strategy [24, 19]. First, we define a union set of modalities in the source and target task, $M = M^S \cup M^T$. We use the input features from all modalities in M when training a policy in the source or target task. If a modality is missing, we replace its features with a vector of zeros. This strategy allows us to have the same representation of the state in the source and target task, and we can apply the proposed adaptation algorithm. However, if there is a modality in the target task that does not exist in the source task (situation 1 and 3 above), the source task's policy cannot handle features from this modality. As a result, the adapted policy will not be optimal, since it also does not know how to handle unseen information from modalities that do not appear in the source task. In summary, if the adaptation setting falls into either the first or the third situation, the proposed method cannot guarantee to learn a good policy. However, since all necessary information in the target task is included in the source task, the adapted policy can select appropriate actions in the second situation. Note that in the second situation, a simpler strategy is to discard all the features from modalities in the source task that does not appear in the target task, then use the remaining features as input for training a policy for the source task. However, this strategy limits the generalization of the source task's policy. When we need to perform adaptation with different target task, the source task's policy needs to be trained again.

When performing policy adaptation, it is also desirable if we can determine the similarities between the source and target tasks. From the multi-modality viewpoint, if the adaptation setting falls into situations 1 or 3 above, then we can say that the source and target tasks are dissimilar. Thus, the policy learned by DPRA cannot guarantee to have good performance. On the other hand, if the adaptation setting is the second situation or an ideal situation ($M^T \equiv M^S$) then the DPRA can learn a good policy for the target task.

While in this research, I use the health consultation task (negotiation) as a use case for the study about multimodal dialog management, the proposed DPRA is flexible and can be applied for any reinforcement-learning-based dialog policy adaptation. In principle, if we want to adapt from a source to a target task that satisfies Assumption 1 and 2, then DPRA can be used. For example, we can use DPRA to adapt policies for visual question-answering tasks where the source and target tasks have a different set of questions that the system can make. In addition to that, we can use DPRA for policy adaptation in both modular-based and end-to-end dialog management. Since the state representation in a modular-based approach usually has a lower number of general dimensions, I expect that the adaptation of modular-based polices will be easier than end-to-end policies. However, with modular-based policy adaptation, the training data must include high-level multimodal labels such as deception or emotion. On the other hand, adaptation for end-to-end policies does not have this requirement, and the amount of data for adaptation can be much larger.

5.3 Evaluation

5.3.1 Setting

I conduct experiments to assess the following hypotheses:

1. The proposed adaptation algorithm, PDRA, requires much less training time than the conventional fine-tuning methods.
2. PDRA achieves equivalent or higher performance compared to current methods in the case of limited data available in the target task.

For comparison with the proposed methods, I use policy adaptation by action embedding [29]. This method uses the same network of the end-to-end dialog modeling and changes the last part of the network for connecting to a new action space in the target task. This model is straightforward but does not require any prior information, such as relations between actions. The proposed algorithm also does not require such prior information; thus, I select this method as the baseline.

In the evaluation, I perform policy adaptation for multimodal goal-oriented dialog system with end-to-end approach.As discussed in [43], the available corpora for this type of conversation are mostly in small-scale; thus, it is suitable for the assessment of the second hypothesis. I use similar end-to-end dialog model as shown in Figure 15, with a multimodal fusion component that uses *Hierarchical Tensor Fusion Network*.

Like the previous chapter, I formulate the problem of dialog management with the episodic and undiscounted reward setting and train the dialog policies using REINFORCE [53] with the state value function as a baseline.

The output layers of the multimodal fusion component and the DST both have 128 units. Therefore, the dialog state is represented by a vector $\mathbf{s} \in \mathcal{R}^{128}$. The neural network representing the dialog policy has one single hidden layer with a dimension of 128, which is fully connected to the input and output layer. I use Adam optimizer to optimize the networks' parameters, and the learning rate is initialized at $1e-3$. I train the policies for source and target tasks with 20,000 and 10,000 episodes respectively, each episode can be seen as a simulated dialog with the user simulator. The learning rate decreases by 10% every 1000 episodes.

The neural networks that approximate the mixture variables in Algorithm 1 and Algorithm 2 have one hidden layer with 256 units. I also use Adam optimizer for parameter optimization, the learning rate is fixed at $1e-4$, and the number of training epochs is 10. Note that in Algorithm 1 and Algorithm 2, the network parameters are updated sequentially with two different gradients. Thus, the training process may have large oscillation and converges slowly. To avoid this problem, I adopt $p : q$ training scheme. Every epoch, I perform component matching for $p = 2$ times and train the mixture model for $q = 2$ times.

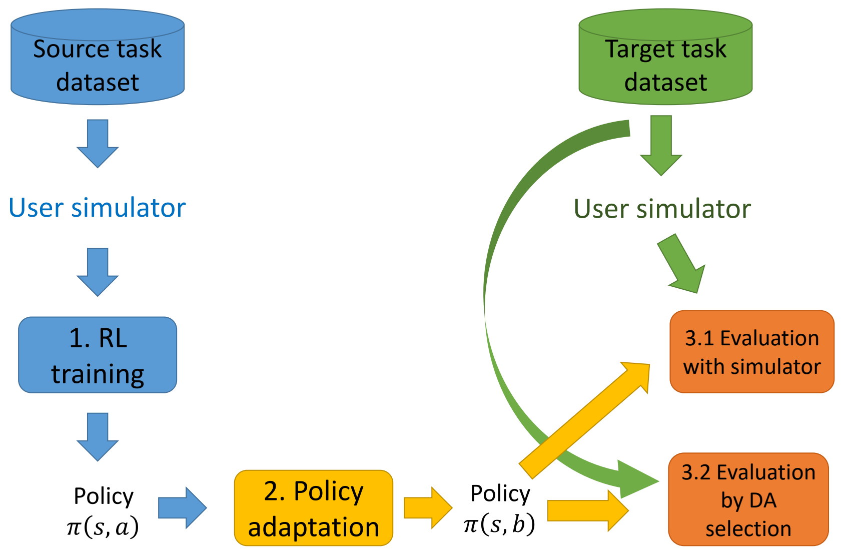

Figure 18. Experiment procedure of policy adaptation evaluation. Each phase is illustrated with different colors and numbering.

I perform evaluation with two settings: similar tasks adaptation and distinctive tasks adaptation, which are explained in Section 5.3.2 and 5.3.3.

5.3.2 Policy adaptation between similar tasks

In this evaluation setting, both the source and the target task are negotiation in the healthcare domain. In particular, the system tries to convince the user that their current living style is unhealthy, and the user needs to adopt a living habit proposed by the system.

The healthcare consultation task described in Chapter 2 is chosen as the source task. I use the healthcare consultation dataset in [34], which contains conversations of six topics: sleeping, eating, working, exercising, social media usage, and leisure activities. The conversations of the first four topics (51 dialogs in total) are used for training the policy of source task. The remaining 24 dialogs in two topics are used for training in the target task. In each turn, I split the recorded video of the user into 30 segments, and sample one frame randomly from each segment. I use the OpenFace toolkit [1] to extract 14 face action unit (AU) regressions and 6 AU classification values for each frame. The visual observation at each turn o^v is the vector that contains these extracted values. Acoustic features are extracted from the audio using the OpenSMILE toolkit [14] with the Interspeech 2009 (IS09) emotion challenge standard feature-set as the feature template [42]. I use these extracted features to create the acoustic observation o^a. Table 9 shows the reward in the source task that the agent receives when selecting an action given the user dialog act (u) and deception label (d), which are generated by the user simulator. The reward function for the source task is defined in Table 9.

Table 9. Reward definition for the source task.

Dialog state		Rewards		
User DA (u)	d	Offer	Framing	End
Accept	0	–10	–10	+100
	1	–10	+10	-100
Reject	0	+10	+10	–100
	1	–10	+10	–100
Question	0	–10	+10	–100
Hesitate	0	+10	+10	–100

In order to make differences between the target and the source task, I change the system's action set of the target task into {*Offer_New, Offer_Change, Framing_Argue, Framing_Answer, End_Dialog*}. Therefore, the source policy never sees these actions during training. The new reward definition is shown in Table 10. *Offer_New* gives +10 reward only if it is selected in the first turn. I use the same multimodel user simulation as described in Chapter 4 for the training of policies in this evaluation.

Table 10. Reward definition for the target task.

Dialog state		Rewards				
User DA (u)	d	Offer_Change	Offer_New	Framing_Answer	Framing_Argue	End
Accept	0	–10	–10	–10	–10	+100
	1	–10	–10	–10	+10	-100
Reject	0	+10	–10	–10	–10	–100
	1	–10	–10	–10	+10	–100
Question	0	–10	–10	+10	–10	–100
Hesitate	0	+10	–10	–10	+10	–100

Table 11. Comparison of training time required for different policy adaptation methods.

Model	Training time
DPRA - regression component matching	$\sim$ 40s
DPRA - MDN component matching	$\sim$ 45s
Policy adaptation by action embedding [29]	$\sim$ 350s

Table 11 shows the training time required for all policy adaptation methods. The numbers reported for DPRA are from the case in which all 24 dialogs of the target task are available for training. For the cases where we have a smaller amount of data, it will take less time for training with DPRA. With policy adaptation by action embedding [29], I choose the number of episodes (interactions with simulated user) for training the target policy based on average rewards received per episode during learning. As can be seen from table 11, PDRA requires significantly less time for training; thus, the first hypothesis stands.

I recreate a scenario where the amount of data available for training in the target task is limited to assess the second hypothesis. In particular, I sample k dialogs from the source task dataset, $k \in \{1, 2, 4, 8, 16\}$. After that, I use these k dialog samples to create the user simulator to train the target policy in *ActEmb* adaptation method and modeling the action-relation probability in PDRA. For each value of k, I sample k dialogs five times, thus making five

Table 12. Average reward per episode of the learned policies with different amounts of available data. Numbers in bracket indicates 95% confidence interval.

Model \ # available dialogs	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs
DPRA - MDN	-25.43 (±16.16)	-10.33 (±8.94)	-2.18 (±10.71)	**14.93 (±4.11)**	17.21 (±4.2)
DPRA - regression	**-12.96 (±11.05)**	**-0.01 (±10.24)**	**6.11 (±6.03)**	14.65 (±6.27)	**19.41 (±4.49)**
ActEmb - 2k	-33.56 (±17.03)	-11.48 (±10.33)	0.01 (±11.45)	-8.56 (±13.00)	12.86 (±11.05)
ActEmb - 10k	-29.98 (±13.12)	-18.07 (±11.60)	-13.42 (±12.22)	0.64 (±11.99)	2.04 (±13.72)
NoAdapt - 2k	-52.93 (±16.59)	-23.42 (±13.93)	-26.61 (±17.98)	-19.15 (±18.29)	-11.77 (±19.80)
NoAdapt - 10k	-36.95 (±15.21)	-17.72 (±16.27)	-9.02 (±13.92)	-7.75 (±14.40)	10.55 (±19.95)

different datasets. With each dataset, I perform policy adaptation ten times for each method. Therefore, I conducted 50 runs of the policy adaptation experiment with each sampling of k dialogs.

Table 12 shows the performance of dialog policies in terms of average reward per episode. The details of reward function for both the source and target task are provided in Appendix (ref here). DPRA-MDN and DPRA-regression refer to the proposed policy adaptation method with component matching by MDN and regression respectively. ActEmb-2k and ActEmb-10k refers to policy adaptation by action embedding method, with the number of episodes for training in the target task is 2,000 and 10,000. Finally, NoAdapt refers to the policy that is trained on the source task without adaptation, the notation for number of training episodes is the same as ActEmb. As can be seen, in general, the performance goes higher with more data available. In Table 12, bold numbers indicate the policy with highest average reward per episode in each scenario of k dialogs available. We can see that DPRA shows significantly improvement in comparison

Table 13. Dialog act selection accuracy of the learned policies with different amount of available data. Numbers in bracket indicates 95% confidence interval.

Model \ # available dialogs	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs
DPRA - MDN	40.96% (±7.79%)	39.86% (±6.81%)	56.16% (±7.82%)	62.05% (±4.94%)	62.05% (±4.64%)
DPRA - regression	**45.41% (±5.99%)**	**52.79% (±5.18%)**	**60.48% (±6.02%)**	**62.98% (±5.08%)**	**69.30% (±3.22%)**
ActEmb - 2k	33.83% (±4.88%)	33.25% (±4.73%)	41.41% (±5.92%)	39.41% (±5.75%)	43.64% (±5.41%)
ActEmb - 10k	28.04% (±5.24%)	24.79% (±5.04%)	26.11% (±6.78%)	25.62% (±6.06%)	29.30% (±5.04%)
NoAdapt - 2k	33.80% (±7.54%)	35.86% (±7.06%)	35.62% (±7.06%)	34.80% (±7.17%)	43.14% (±7.50%)
NoAdapt - 10k	37.39% (±6.91%)	44.50% (±8.01%)	47.86% (±7.61%)	44.41% (±6.31%)	52.16% (±7.76%)

with ActEmb and NoAdapt ($p < 0.05$), especially when k is small. We can also observe that ActEmb-2k and ActEmb-10k performs similarly; on the other hand, the performance of NoAdapt-10k is remarkably higher than NoAdapt-2k.

The performance in terms of dialog act selection accuracy for all policies is shown in Table 13. Similarly, the dialog policies learned by PDRA outperform those from ActEmb and NoAdapt with a large margin ($p < 0.05$). Surprisingly, with more data available, the performance gap between DPRA and the other methods becomes bigger. There is only a subtle increase in action selection accuracy of the policies learned by ActEmb and NoAdapt when k increases from 1 to 16. Let us recall that the results in Table 12 are reported under the setting where I run the policies are against a simulator that is created from all 24 dialogs in the target dataset. Therefore, if we train a policy by using a simulator created form 16 dialogs, the performance in terms of average reward per episode will be much higher than using the simulator from one dialog. However, because ActEmbed and NoAdapt do not use full knowledge from the source task policy, 16 dialogs are not enough to train a policy with high action selection accuracy. Thus, the gain when increasing the amount of available data in Table 13 is modest. In contrast, DPRA can retain knowledge from the source task policy and effectively adapts it to the target task, thus achieving high performance in terms of action selection accuracy, especially when more data is available.

Table 14. Average reward per episode of the policies that take only use linguistic features (single-modality). Numbers in bracket indicates 95% confidence interval.

# available dialogs / Model	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs
DPRA - MDN	**-16.61 (±5.79)**	-15.61 (±15.43)	-6.60 (±6.71)	-2.15 (±5.60)	**8.07 (±5.27)**
DPRA - regression	-19.98 (±14.07)	**-8.59 (±8.81)**	**2.53** (±7.01)	**2.71 (±8.19)**	3.52 (±5.46)
ActEmb - 2k	-33.49 (±12.02)	-23.78 (±10.27)	-19.40 (±13.08)	-15.66 (±14.22)	-8.45 (±14.11)
ActEmb - 10k	-41.42 (±14.56)	-16.78 (±9.92)	-18.97 (±14.765)	1.69 (±11.85)	0.65 (±18.98)
NoAdapt - 2k	-47.47 (±13.77)	-44.22 (±14.51)	-40.98 (±16.31)	-28.17 (±11.94)	-23.95 (±19.42)
NoAdapt - 10k	-45.71 (±13.09)	-34.14 (±17.69)	-23.81 (±15.53)	-12.86 (±16.52)	-11.35 (±22.79)

In addition to experiments with multimodal dialog policies, I also conduct adaptation experiments for adaptation in the case of single-modality dialog management. In this setting, the dialog managers select actions based on linguistic features only. I use exactly the same setup used in the previous experiment with multimodal dialog management. For each sampling of k dialogs as the training dataset, I run the experiments for five times. Table 14 and 15 show the results in terms of average rewards received per episode and DA selection accuracy. As can be seen, policies adapted by DPRA achieves best performance in both metrics. In terms of average reward, DPRA significantly outperforms other adaptation method when $k \in 1, 2, 4$ with $p < 0.05$. In terms of DA selection, the proposed method gets significantly higher performance with all values of k except for $k = 1$.

Another notable point is that performance in terms of DA selection decreases remarkably when the dialog managers use features from only one modality. This phenomenon proves that for the health consultation task, multimodality information is important.

Table 15. Dialog act selection accuracy of the learned policies that only use linguistic features (single-modality). Numbers in bracket indicates 95% confidence interval.

Model \ # available dialogs	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs
DPRA - MDN	**38.77% (±5.24%)**	**41.29% (±4.48%)**	**47.73% (±2.57%)**	**49.92% (±2.09%)**	**51.24% (±4.21%)**
DPRA - regression	30.98% (±8.07%)	37.02% (±6.61%)	46.16% (±5.49%)	44.72% (±4.27%)	50.74% (±4.24%)
ActEmb - 2k	33.11% (±6.86%)	32.10% (±6.51%)	35.37% (±5.34%)	37.99% (±5.75%)	46.05% (±6.01%)
ActEmb - 10k	36.37% (±5.20%)	38.88% (±5.65%)	42.57% (±5.59%)	44.37% (±6.39%)	45.25% (±6.36%)
NoAdapt - 2k	32.55% (±4.54%)	29.02% (±4.99%)	30.06% (±6.79%)	33.27% (±4.63%)	43.00% (±6.62%)
NoAdapt - 10k	33.54% (±5.04%)	39.90% (±6.14%)	44.05% (±5.12%)	43.80% (±5.15%)	46.35% (±6.93%)

5.3.3 Policy adaptation between distinctive tasks

With this evaluation, I choose the negotiation task described previously as the source task. The target task is positive emotion elicitation, where the system needs to evoke positive emotions from the user and makes them feel better.

Similar to the previous evaluation, I use the healthcare consultation corpus in [34] as the source task dataset. In this case, I use all of the data (75 dialogs) to train the source task's policy. For the target task, I use the emotion elicitation dialog corpus [26, 27], which contains 58 conversations in total. Each conversation corresponds to a session where a participant watches an emotion-induced video and then proceeds to talk with an expert, a professional counselor. The goal of the expert is to elicit positive emotions from the participant during this conversation. Table 16 shows some statistics of the emotion elicitation dataset.

Table 16. Emotion elicitation dataset statistics.

Total number of sessions	58
Total duration of all sessions	23 hours 41 minutes
Average duration per session	23.6 minutes
Minimum duration	10 minutes
Maximum duration	33 minutes

In the emotion elicitation task, the user simulator is built similarly to the one described in Section 4.3, with the change into emotion model instead of deception model,

$$\begin{aligned} \text{intention model} &= P(u_{t+1}|u_t, e_t, e_{t+1}, a_t) \\ \text{emotion model} &= P(e_{t+1}|u_t, e_t, a_t) \end{aligned} \tag{25}$$

The emotion label in Equation 25 is decided based on the annotated valence values, which are quantized into $\{0, 1, 2, 3, 4\}$ from the range $[-1, 1]$. Since the conversations in emotion elicitation tasks are much longer, the training of a policy for this task is also more difficult. Therefore, I increase the number of episodes for policy learning from 10,000 to 25,000. Besides, videos in the emotion dataset are of insufficient quality for facial expression extraction. As a result, the dialog manager does not take visual observation as input in the target task. When using the dialog manager's network from the source task to perform state tracking in the target task, visual observation is replaced by a constant vector with all values are equal to one. All other experiment settings are similar to description in Section 5.3.1.

This adaptation setting does not follow the assumptions raised in Section 5.2, because the source and the target task has little similarities in both state representation and the state space. Let us recall that we use the dialog manager in the source task to perform state tracking and get the dialog samples for policy adaptation by DPRA. This strategy ensures that the state representation is the same between the source and the target task. However, the resultant of DST of the source task's dialog manager is a state that does not contain emotion information, since the user simulator does not have it. On the other hand, in the target task, emotion information is essential and should be included in state representation. Therefore, I expect that the policies that are adapted by DPRA do not work well in this setting.

Table 17. Average reward per episode of the learned policies with different amount of available data. Numbers in bracket indicates 95% confidence interval.

# available dialogs / Model	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs	32dialogs
DPRA - MDN	32.15 (±9.05)	29.47 (±9.90)	34.65 (±12.23)	13.13 (±4.90)	23.49 (±10.53)	24.90 (±9.13)
DPRA - regression	38.73 (±10.94)	21.57 (±9.47)	31.32 (±7.15)	18.61 (±7.45)	1.46 (±2.85)	32.56 (±10.48)
ActEmb - 5k	48.08 (±8.08)	33.31 (±13.84)	**46.13 (±10.89)**	5.61 (±38.33)	**79.72 (±36.50)**	44.30 (±15.78)
ActEmb - 25k	42.51) (±13.45)	**37.43 (±11.44)**	30.21 (±13.68)	45.35 (±9.85)	48.84 (±13.78)	**86.67 (±42.12)**
NoAdapt - 5k	33.97 (±13.62)	31.06 (±11.56)	36.36 (±10.17)	**45.96 (±23.65)**	38.65 (±7.09)	45.16 (±10.15)
NoAdapt - 25k	**66.53 (±44.95)**	26.47 (±11.84)	35.05 (±11.71)	71.84 (±45.52)	53.06 (±11.75)	72.78 (±18.37)

Table 17 shows the average of total reward per episode of the policies adapted for the emotion elicitation task. As expected, since there are few similarities between the source and the target task in this setting, the proposed method does not work as well as ActEmb or NoAdapt. The adapted policies have lower performance in terms of total reward per episode. However, we can still see that policies adapted by DPRA gain positive rewards in all scenarios, which indicates a small connection between the health consultation task and the emotion elicitation task.

In Table 18, we can see that in terms of DA selection accuracy, there are not much difference between the policies learned by the proposed method and previous methods. Overall, the performance of policies trained by adaptation methods (DPRA and ActEmbed) are not higher than those trained without adaptation (NoAdapt).

Table 18. Dialog act selection accuracy of the learned policies with different amount of available data. Numbers in bracket indicates 95% confidence interval.

# available dialogs / Model	1 dialog	2 dialogs	4 dialogs	8 dialogs	16 dialogs	32 dialogs
DPRA - MDN	21.94% (±5.35%)	24.79% (±6.59%)	24.03% (±6.12%)	30.14% (±4.07%)	19.27% (±4.84%)	**31.09% (±5.64%)**
DPRA - regression	23.79% (±7.40%)	28.46% (±7.07%)	23.33% (±5.45%)	29.59% (±7.31%)	**42.37% (±2.19%)**	26.59% (±5.27%)
ActEmb - 5k	31.49% (±4.48%)	28.12% (±7.21%)	37.53% (±5.05%)	32.91% (±6.50%)	23.22% (±7.10%)	23.00% (±8.49%)
ActEmb - 25k	**35.19% (±4.30%)**	**28.91% (±7.38%)**	**40.29% (±5.50%)**	34.62% (±6.18%)	22.07% (±6.92%)	21.23% (±7.84%)
NoAdapt - 5k	29.75% (±6.54%)	28.29% (±6.87%)	25.55% (±8.20%)	31.99% (±7.53%)	18.85% (±7.25%)	27.24% (±7.60%)
NoAdapt - 25k	31.11% (±7.39%)	19.38% (±7.03%)	28.46% (±8.63%)	**35.54% (±5.40%)**	20.21% (±6.96%)	30.26% (±6.58%)

6. Conclusions and future directions

6.1 Conclusions

This research studied the application of the end-to-end approach for multimodal goal-oriented dialog management. This thesis tackled two major challenges of this topic: fusion of multimodal information and data sparsity. I also conducted a study that compares modular-based and end-to-end dialog management.

This research introduces a new method called hierarchical tensor fusion network for combining multiple modalities. Unlike previous works, the proposed method can balance the difference between modalities and learn about feature interactions simultaneously, which allows us to create better multimodal processing models. Experimental results indicated that the hierarchical tensor fusion model has the best performance, outperforming the existing approaches used in previous studies (hierarchical and tensor fusion) in the deception detection and sentiment analysis tasks. I also investigated in experiments about the DA selection accuracy of a negotiation dialog system. I found out that the dialog system achieves better performance when using labels from the deception detection model that uses hierarchical tensor fusion.

For the problem of data sparsity, I proposed a novel method for policy adaptation in dialog management, which is termed *dialog policy reuse algorithm – DPRA*. The proposed method uses the action-relation probability for adaptation, which allows reusing of the source task policy for action selection of the target task. DPRA learns the action-relation probability from the dialog samples in both tasks using mixture density network, and is able to immediately derive a policy for the target task. Thus, DPRA can learn the target task's policy in a much shorter time in comparison to conventional methods that require RL training and user simulation. Since the proposed method does not use the user simulator, the problem of low performance due to errors in constructing the user simulator can be avoided. Experimental results showed that the policy learned by DPRA performs better than those adapted by previous adaptation methods when the amount of available data is small and the source and target tasks are similar.

6.2 Future directions

Regarding the future direction of hierarchical tensor fusion, the major drawback of the proposed and current methods is that they require a manual design of the network structure, which is tedious and requires much effort. A possible solution is to apply the meta-learning method and create a network that can perform structure self-organization. Besides, a combination of hierarchical tensor fusion with the attentional fusion [17] can also help alleviate this problem since the attention weights also balance the modalities without using fully-connected layers.

In terms of policy adaptation, I would like to conduct a scrutiny analysis of DPRA to understand in which adaptation setting it works and what kind of performance we should expect from it. In addition, DPRA does not consider reward dynamics at the moment. I believe that if we can incorporate the estimation of changing reward functions between the source and the target task, we can further improve the performance of the policies learned by DPRA. Finally, I also want to investigate the proposed adaptation method's application to other settings, such as adaptation for autonomous control tasks.

As discussed previously, accurately constructing a multimodal user simulator also affects the dialog manager's performance. In this research, I use a conventional method based on a simple maximum likelihood for building the multimodal user simulator. In the future, I am planning to work on creating simulator with more powerful generative models such as variational auto encoder or generative adversarial networks. Such methods may bring benefit to the training of dialog manager using reinforcement learning.

References

[1] Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. Openface: an open source facial behavior analysis toolkit. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–10. IEEE, 2016.

[2] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[3] Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

[4] Reinaldo AC Bianchi, Luiz A Celiberto Jr, Paulo E Santos, Jackson P Matsuura, and Ramon Lopez de Mantaras. Transferring knowledge as heuristics in reinforcement learning: A case-based approach. *Artificial Intelligence*, 226:102–121, 2015.

[5] Christopher M Bishop. Mixture density networks. 1994.

[6] Lu Chen, Cheng Chang, Zhi Chen, Bowen Tan, Milica Gašić, and Kai Yu. Policy adaptation for deep reinforcement learning-based dialogue management. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6074–6078. IEEE, 2018.

[7] Robert H Crites and Andrew G Barto. An actor/critic algorithm that is equivalent to q-learning. In *Advances in neural information processing systems*, pages 401–408, 1995.

[8] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. Visual dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 326–335, 2017.

[9] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2951–2960, 2017.

[10] Gilles Degottex, John Kane, Thomas Drugman, Tuomo Raitio, and Stefan Scherer. Covarep—a collaborative voice analysis repository for speech technologies. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 960–964. IEEE, 2014.

[11] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmad, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–495, 2017.

[12] Paul Ekman, Wallace V Freisen, and Sonia Ancoli. Facial signs of emotional experience. *Journal of personality and social psychology*, 39(6):1125, 1980.

[13] Mihail Eric and Christopher D Manning. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 468–473, 2017.

[14] Florian Eyben, Martin Wöllmer, and Björn Schuller. Opensmile: the munich versatile and fast open-source audio feature extractor. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1459–1462. ACM, 2010.

[15] Takuya Hiraoka, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Reinforcement learning of cooperative persuasive dialogue policies using framing. In *COLING*, pages 1706–1717, 2014.

[16] Julia Hirschberg, Stefan Benus, Jason M Brenier, Frank Enos, Sarah Friedman, Sarah Gilman, Cynthia Girand, Martin Graciarena, Andreas Kathol, Laura Michaelis, et al. Distinguishing deceptive from non-deceptive speech. In *Interspeech*, pages 1833–1836, 2005.

[17] Chiori Hori, Huda Alamri, Jue Wang, Gordon Wichern, Takaaki Hori, Anoop Cherian, Tim K Marks, Vincent Cartillier, Raphael Gontijo Lopes, Abhishek Das, et al. End-to-end audio visual scene-aware dialog using multimodal

attention-based video features. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2352–2356. IEEE, 2019.

[18] Guosheng Hu, Yang Hua, Yang Yuan, Zhihong Zhang, Zheng Lu, Sankha S Mukherjee, Timothy M Hospedales, Neil Martin Robertson, and Yongxin Yang. Attribute-enhanced face recognition with neural tensor fusion networks. In *Proceedings of ICCV 2017*, 2017.

[19] Vladimir Ilievski, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. Goal-oriented chatbot dialog management bootstrapping with transfer learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4115–4121, 2018.

[20] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in neural information processing systems*, pages 6250–6261, 2017.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR 2015*, 2015.

[22] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[23] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[25] Pierre L'Ecuyer. Efficiency improvement and variance reduction. In *Proceedings of Winter Simulation Conference*, pages 122–132. IEEE, 1994.

[26] Nurul Lubis, Sakriani Sakti, Koichiro Yoshino, and Satoshi Nakamura. Eliciting positive emotion through affect-sensitive dialogue response generation: A neural network approach. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[27] Nurul Lubis, Sakriani Sakti, Koichiro Yoshino, and Satoshi Nakamura. Positive emotion elicitation in chat-based dialogue systems. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):866–877, 2019.

[28] Gideon Mendels, Sarah Ita Levitan, Kai-Zhan Lee, and Julia Hirschberg. Hybrid acoustic-lexical deep learning approach for deception detection. *Proc. Interspeech 2017*, pages 1472–1476, 2017.

[29] Jorge A Mendez, Alborz Geramifard, Mohammad Ghavamzadeh, and Bing Liu. Reinforcement learning of multi-domain dialog policies via action embeddings. In *The 3rd Workshop on Conversational AI: Today' s Practice Tomorrow' s Potential, NeurIPS.*, 2019.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[31] Hosomi Naoki, Sakti Sakriani, Koichiro Yoshino, and Satoshi Nakamura. Deception detection and analysis in spoken dialogues based on fasttext. *Proc. APSIPA 2018*, 2018.

[32] Tung The Nguyen, Koichiro Yoshino, Sakriani Sakti, and Satoshi Nakamura. Impact of deception information on negotiation dialog management. *IWSDS*, 2018.

[33] Tung The NGUYEN, Koichiro YOSHINO, Sakriani SAKTI, Satoshi NAKAMURA, et al. Utilizing deception information for dialog management of doctor-patient conversations. *JSAI 大会論文集*, 2018:2M201–2M201, 2018.

[34] Tung The Nguyen, Koichiro Yoshino, Sakriani Sakti, Satoshi Nakamura, et al. Dialog management of healthcare consulting system by utilizing deceptive information. *Transactions of the Japanese Society for Artificial Intelligence*, 35(1):DSI–C_1, 2020.

[35] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference*

on empirical methods in natural language processing (EMNLP), pages 1532–1543, 2014.

[36] Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, and Mihai Burzo. Deception detection using real-life trial data. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, pages 59–66. ACM, 2015.

[37] Verónica Pérez-Rosas, Mohamed Abouelenien, Rada Mihalcea, Yao Xiao, CJ Linton, and Mihai Burzo. Verbal and nonverbal clues for real-life deception detection. In *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*, pages 2336–2346, 2015.

[38] Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180, 2010.

[39] Alan Ritter, Colin Cherry, and William B Dolan. Data-driven response generation in social media. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 583–593, 2011.

[40] Ognjen Rudovic, Meiru Zhang, Bjorn Schuller, and Rosalind Picard. Multimodal active learning from human data: A deep reinforcement learning approach. In *2019 International Conference on Multimodal Interaction*, pages 6–15, 2019.

[41] Konrad Scheffler and Steve Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*, pages 12–19. Citeseer, 2002.

[42] Björn Schuller, Stefan Steidl, and Anton Batliner. The interspeech 2009 emotion challenge. In *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[43] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49, 2018.

[44] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 553–562, 2015.

[45] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[46] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[47] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640, 2016.

[48] Matthew E Taylor, Gregory Kuhlmann, and Peter Stone. Autonomous transfer for reinforcement learning. In *AAMAS (1)*, pages 283–290. Citeseer, 2008.

[49] Leimin Tian, Johanna Moore, and Catherine Lai. Recognizing emotions in spoken dialogue with hierarchically fused acoustic and lexical features. In *Spoken Language Technology Workshop (SLT), 2016 IEEE*, pages 565–572. IEEE, 2016.

[50] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[51] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[52] Jason D Williams, Pascal Poupart, and Steve Young. Partially observable markov decision processes with continuous observations for dialogue management. In *Proceedings of the 6th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2005.

[53] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[54] James R Wilson. Variance reduction techniques for digital simulation. *American Journal of Mathematical and Management Sciences*, 4(3-4):277–312, 1984.

[55] Koichiro Yoshino and Tatsuya Kawahara. Conversational system for information navigation based on pomdp with user focus tracking. *Computer Speech & Language*, 34(1):275–291, 2015.

[56] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1103–1114. Association for Computational Linguistics, 2017.

[57] AmirAli Bagher Zadeh, Paul Pu Liang, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2236–2246, 2018.

[58] Jiaping Zhang, Tiancheng Zhao, and Zhou Yu. Multimodal hierarchical reinforcement learning policy for task-oriented visual dialog. In *Proceedings of the 19th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 140–150, 2018.

[59] Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 1–10, 2016.

[60] Mingyang Zhou, Josh Arnold, and Zhou Yu. Building task-oriented visual dialog systems through alternative optimization between dialog policy and language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 143–153, 2019.

Appendix

A. Proof of equation 1 and equation 10

First, I show the proof for (15):

$$\begin{aligned}\pi(s,b) &= P(b|s)\\ &= \sum_{a\in A} P(b,a|s) \quad \textit{(law of total probability)}\\ &= \sum_{a\in A} P(b|a,s)P(a|s)\\ &= \sum_{a\in A} P(b|a,s)\pi(s,a) \quad Q.E.D\end{aligned}$$

The full proof for (23) is as follows:

$$\begin{aligned}\sum_{b\in B}\pi(s,b) &= \sum_{b_j\in B} P(b_j|s)\\ &= \sum_{b_j\in B}\sum_{a_i\in A} P(a_i,b_j|s)\\ &= \sum_{b_j\in B}\sum_{a_i\in A} P(a_i|s)P(b_j|a_i,s)\\ &= \sum_{a_i\in A} P(a_i|s)(\sum_{b_j\in B} P(b_j|a_i,s))\\ &= \sum_{a_i\in A} P(a_i|s)(\sum_{b_j\in B} w_{ij}(s))\\ &= \sum_{a_i\in A} P(a_i|s)\cdot 1\\ &\qquad \textit{(sum of component weights is 1)}\\ &= \sum_{a\in A} P(a|s) = 1 \quad Q.E.D\end{aligned}$$

B. Results

The results of Table 12 is shown as a line chart in Figure 19. The error bars indicate 95% confidence interval for each value point.

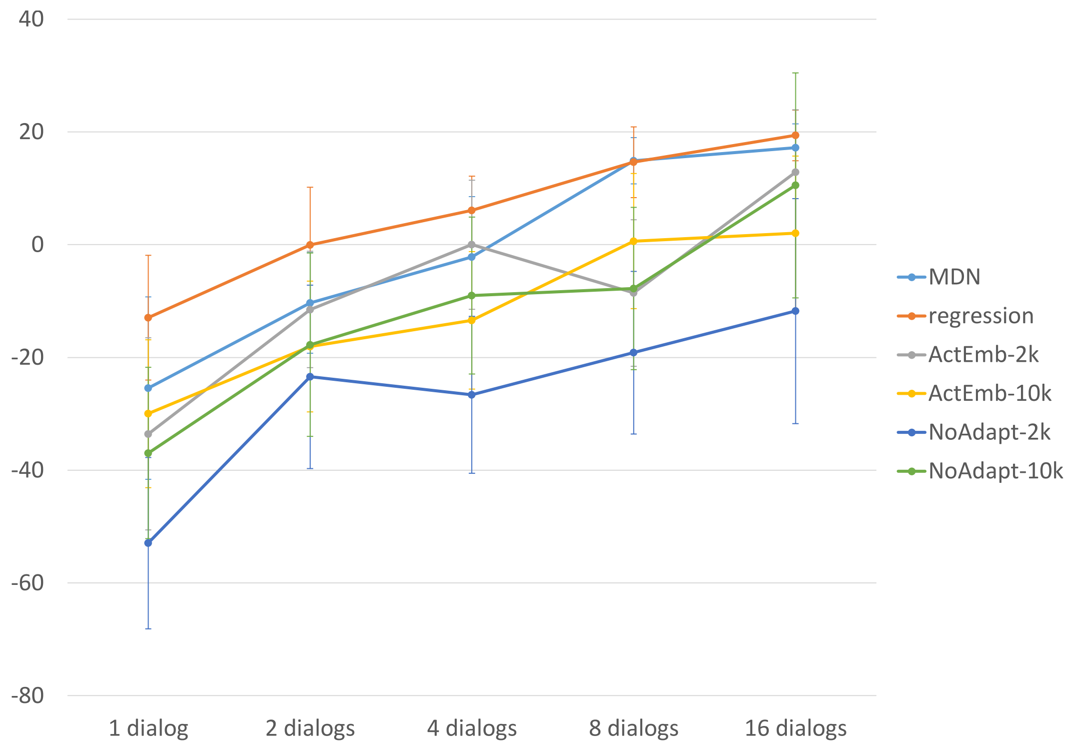

Figure 19. Average reward received per episode of adapted policies in similar tasks setting.

Figure 20 shows the results from Table 13 but in line chart. Similarly, error bars show 95% confidence interval of each value point.

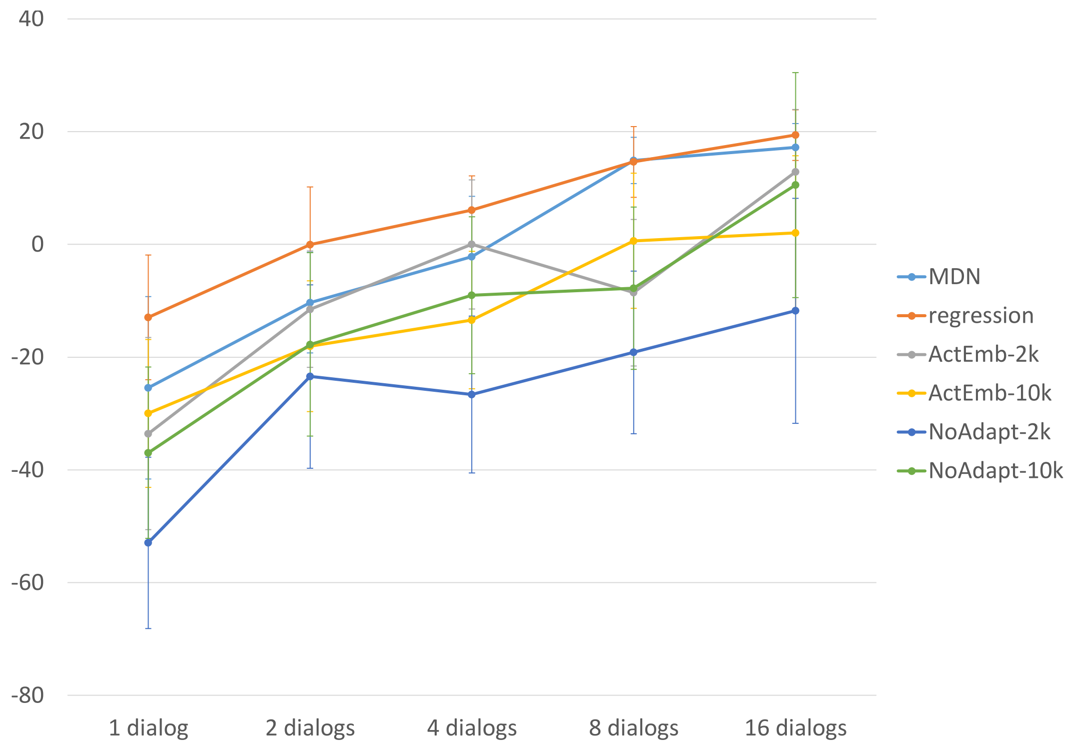

Figure 20. DA selection accuracy received per episode of adapted policies in similar tasks setting.

Figure 21 shows the policies' performance in terms of average total reward per episode in Table 17. The error bars indicate 95% confidence interval for each value point.

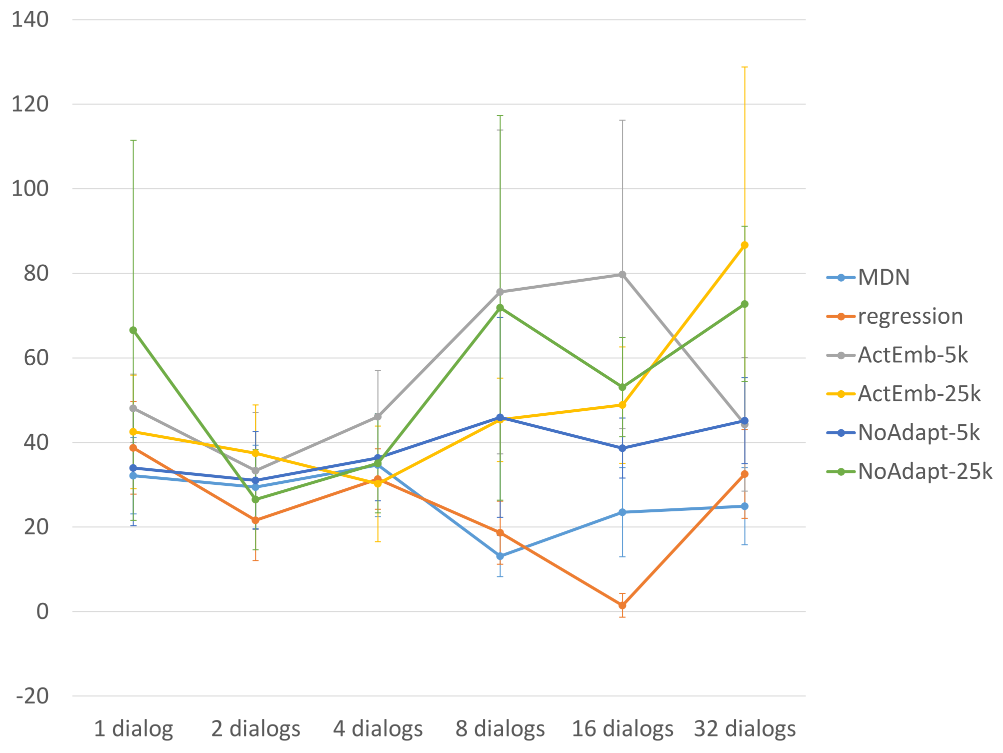

Figure 21. Average reward received per episode of adapted policies in distinctive tasks setting.

Figure 22 displays the results from Table 18 in line chart. Similarly, error bars show 95% confidence interval of each value point.

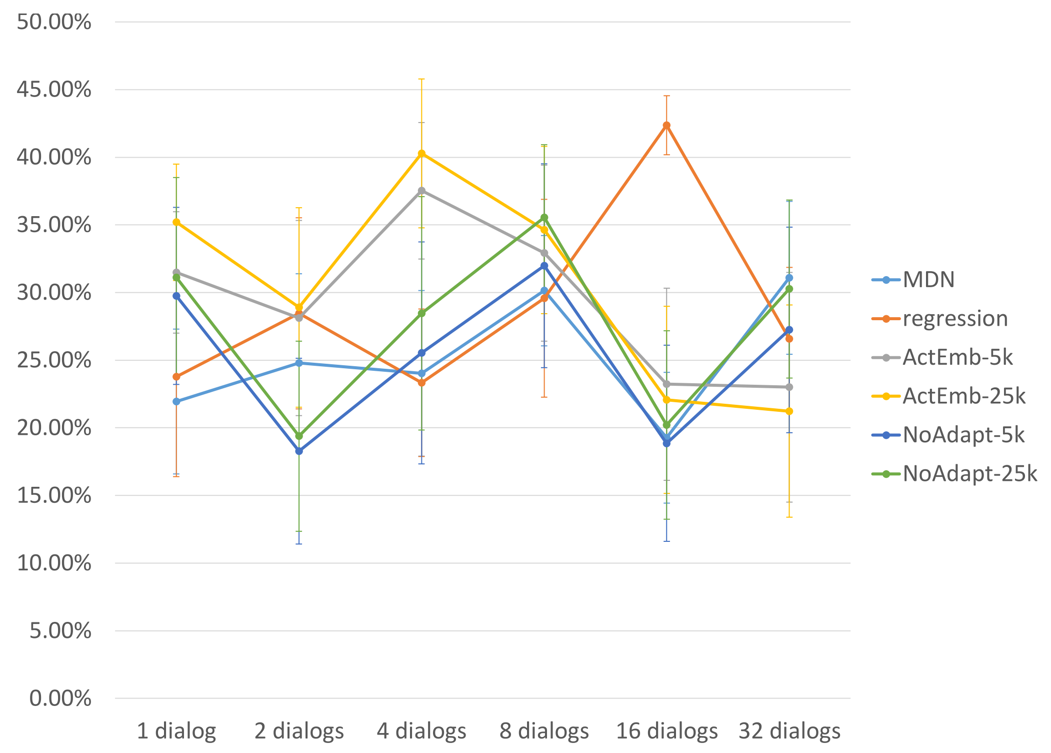

Figure 22. DA selection accuracy received per episode of adapted policies in distinctive tasks setting.

C. Training dialog policy with REINFORCE

This section explains in more details how the policies are trained using REINFORCE. First, I show why the gradient in Equation 13 is equivalent to Equation 12. The right hand side (R.H.S) of Equation 13 can be rewritten as,

$$E\left[\sum_{t=0}^{T} R_t \frac{\partial log\pi^{\theta}(s_t, a_t)}{\partial\theta}\middle|\pi\right] - E\left[\sum_{t=0}^{T} b(s_t) \frac{\partial log\pi^{\theta}(s_t, a_t)}{\partial\theta}\middle|\pi\right] \tag{26}$$

We can derive the right component of this equation as follows:

$$
\begin{aligned}
& E\left[\sum_{t=0}^{T} b(s_t)\frac{\partial log\pi^\theta(s_t,a_t)}{\partial\theta}\middle|\pi\right] \\
&= \sum_{a_t}\pi^\theta(s_t,a_t)\sum_{t=0}^{T} b(s_t)\frac{\partial log\pi^\theta(s_t,a_t)}{\partial\theta} \\
&= \sum_{t=0}^{T} b(s_t)\sum_{a_t}\frac{\partial\pi^\theta(s_t,a_t)}{\pi^\theta(s_t,a_t)\partial\theta} \\
&= \sum_{t=0}^{T} b(s_t)\sum_{a_t}\pi^\theta(s_t,a_t)\frac{\partial\pi^\theta(s_t,a_t)}{\pi^\theta(s_t,a_t)\partial\theta} \\
&= \sum_{t=0}^{T} b(s_t)\frac{\partial\sum_{a_t}\pi^\theta(s_t,a_t)}{\partial\theta} \\
&= \sum_{t=0}^{T} b(s_t)\cdot 0 \qquad (since\sum_{a_t}\pi^\theta(s_t,a_t)=1) \\
&= 0
\end{aligned}
$$

Replace this result into Equation 26, we can see that introducing the baseline $b(s_t)$ into Equation 12 does not change the gradient.

Equation 14 shows the true gradient that is used for updating policy parameters θ. However, the baseline $V^\theta(s_t)$ cannot be estimated without bias in the case of one-sample update. [46] provide a solution for this problem using function approximation. In principle, we use a function $f^w(s)$ with parameter w to estimate $V^\theta(s)$. This function can be trained using mean-squared-error on the sampled return.

$$L_{MSE} = \frac{1}{2}(R_t - f^w(s_t))^2 \tag{27}$$

The update gradient for policy training by REINFORCE is then given by,

$$\partial(\theta) = E\left[\sum_{t=0}^{T}(R_t - f^w(s_t))\frac{\partial log\pi^\theta(s_t,a_t)}{\partial\theta}\middle|\pi\right] \tag{28}$$

Publication list

1. ”Impact of deception information on negotiation dialog: a use case on doctor-patient conversation,” Tung The Nguyen, Koichiro Yoshino, Sakti Sakriani and Satoshi Nakamura, in International Workshop on Spoken Dialogue System Technology (IWSDS2018), Singapore, Singapore, 2018

2. ”Utilizing deception information for dialog management of doctor-patient conversations,” Tung The, NGUYEN, YOSHINO, Koichiro, Sakriani SAKTI, and Satoshi NAKAMURA, in The 32th Annual Conference of the Japanese Society for Artificial Intelligence, 人工知能学会全国大会論文集 第 32 回全国大会 (2018). 一般社団法人 人工知能学会, 2018.

3. ”Hierarchical Tensor Fusion Network for Deception Handling Negotiation Dialog Model,” Tung, The Nguyen, Yoshino, Koichiro, Sakriani Sakti, and Satoshi Nakamura, in The 3rd conversational AI workshop – NeurIPS 2019.

4. ”Dialog management of healthcare consulting system by utilizing deceptive information,” Tung The Nguyen, Yoshino Koichiro, Sakti Sakriani and Satoshi Nakamura, in Transactions of the Japanese Society for Artificial Intelligence, 35(1):DSI–C 1, 2020.

5. ”Policy reuse for dialog management using action-relation probability,” Tung, The Nguyen, Yoshino, Koichiro, Sakriani Sakti, and Satoshi Nakamura, in IEEE Access, vol. 8, pp. 159639-159649, 2020.