UNIVERSITY OF CALIFORNIA,
IRVINE

Learning Graph Structures in Task-Oriented Dialogue Systems

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Engineering

by

Jie Wu

2021

DEDICATION

To my mom, Kunying Guo, for her endless encouragement and support
To my wife, Jiwei Deng, and my daughter, Olivia, for their love

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Professor Ian G. Harris for his guidance, and support throughout my Ph.D. study. He has been encouraging me to explore this fascinating field, natural language processing, guiding me to overcome various difficulties in my research and inspiring me to come up with novel ideas. I have learnt from Professor Harris not only the academic knowledge, but also his working philosophy, being down-to-earth and hard-working. I still remembered that he arduously reviewed my paper over the weekend in order to make a hard deadline. Without his patience and encouragement hardly can I complete this work. I can't describe enough how thankful I am for his support over the years.

I own my sincere gratitude to Professor Nader Bagherzadeh, for leading me to the Department of Electrical Engineering and Computer Science at the University of California, Irvine. It has been a wonderful experience to have my Ph.D. study in UC Irvine. I also would like to acknowledge Professor Jean-Luc Gaudiot and Professor Chen-Yu (Phillip) Sheu for serving as my doctoral committee members. They take their valuable time to join my Ph.D. exams and defense for showing their great support and providing their insightful feedback.

I am deeply grateful to my outstanding colleague and collaborators for their insightful contribution and feedback to my research work. I would like to extend my gratitude to Professor Hongzhi Zhao at the Beijing Jiaotong University for his collaboration in a variety of projects. I am also thankful to my co-worker Dr. Min Soo Kim for his great help.

My graduate work and this thesis have been supported by Qualcomm, Microsoft and Salesforce. Their great employee tuition assistance program helps full-time employees like me to complete my Ph.D. degree and I feel grateful and proud to work for these companies.

VITA

Jie Wu

EDUCATION

Doctor of Philosophy in Computer Engineering	**2021**
University of California, Irvine	*Irvine, California*
Master of Science in Electrical Engineering	**2009**
Michigan Technological University	*Houghton, Michigan*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2014–2021**
University of California, Irvine	*Irvine, California*

PROFESSIONAL EXPERIENCE

Lead Software Engineer	**2019–Current**
Salesforce	*Kirkland, Washington*
Senior Software Engineer	**2016–2019**
Microsoft	*Redmond, Washington*
Senior Software Engineer	**2012–2016**
Qualcomm	*San Deigo, Californina*

PUBLICATIONS

Jie Wu, Ian G. Harris, Hongzhi Zhao. “Spoken Language Understanding for Task-oriented Dialogue Systems with Augmented Memory Networks.” Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. (NAACL) 2021:797-806.

Jie Wu, Ian G. Harris, Hongzhi Zhao. “GraphMemDialog: Optimizing End-to-End Task-Oriented Dialog Systems Using Graph Memory Networks.” Submitted to the Association for the Advancement of Artificial Intelligence (AAAI) 2022.

Jie Wu, Ian G. Harris, Hongzhi Zhao, Guangming Ling. “A Graph-to-Sequence Model for Joint Intent Detection and Slot Filling in Task-Oriented Dialogue Systems.” Submitted to Proceedings of ACL 2022.

Guangming Ling, Aiping Xu, Chao Wang and **Jie Wu**. “REBDT: A regular expression boundary-based decision tree model for Chinese logistics address segmentation in IoT.” Submitted to Applied Intelligence.

Hongzhi Zhao, Fang Zhang, Baosheng Wang, **Jie Wu** and Nader Bagherzadeh. “A memory access scheduling method for DNN training processor.” Submitted to Microprocessors and Microsystems: Embedded Hardware Design.

Hongzhi Zhao, Nader Bagherzadeh, **Jie Wu**. “A general fault-tolerant minimal routing for mesh architectures.” IEEE Transactions on Computers. 2017, 66(7): 1240-1246.

ABSTRACT OF THE DISSERTATION

Learning Graph Structures in Task-Oriented Dialogue Systems

By

Jie Wu

Doctor of Philosophy in Computer Engineering

University of California, Irvine, 2021

Professor Ian G. Harris, Chair

Dialogue systems, also known as conversational agents, are intelligent computer systems which converse with a human via natural language (text, or speech). They generally fall into two categories based on their applications, that is, chit-chat dialogue systems (chatbots) and task-oriented dialogue systems. The former is primarily interacting with humans to provide reasonable responses and entertainment on open domains, whereas the latter aims to help users complete a dedicated task on one specific domain, for example, inquiring about weather, reserving restaurants, or booking flights. This thesis focuses primarily on task-oriented dialogue systems.

The earliest dialogue systems were highly dependent on complicated and hand-crafted rules and logics, which were costly and unscalable. With the recently unprecedented progress in natural language processing propelled by deep learning, statistical dialogue systems have gradually become well used for reducing cost and providing robustness. Traditionally, these systems were highly modularized in a pipelined manner, including spoken language understanding, dialogue state tracking, dialogue management, and response generation. However, modularized design typically requires expensive human labeling and easily leads to error propagation due to module dependencies. On the other hand, end-to-end neural-based models learn the dialogue hidden representation automatically and generate system responses directly, thereby requiring much less human effort and being more scalable to new domains. Although recurrent neural network (RNN) based neural models show promising results, they still suffer from several major drawbacks. First, dominant RNNs are inherently unstable over long-time sequences as RNNs tend to focus more on short-term memories and forcefully compress historical records into one hidden state vector (Weston et al., 2014). Second, RNNs focus primarily on modeling sequential dependencies, and thus rich graph structure information hidden in the dialogue context is completely ignored. Lastly, effectively incorporating external knowledge into end-to-end task-oriented dialog systems still remains a challenge.

In this thesis, we are dedicated to addressing these limitations of conventional neural models and propose end-to-end learning frameworks to model long-term dialogue context and to learn dialogue graph structures. In addressing the weakness of RNNs processing long sequences, we propose a novel approach to model long-term slot context and to fully utilize the semantic correlation between slots and intents. We adopt a key-value memory network to model slot context dynamically and to track more important slot tags decoded before, which are then fed into our decoder for slot tagging. Furthermore, gated memory information is utilized to perform intent detection, mutually improving both tasks through global optimization. We empirically show that our key-value memory networks enable effectively tracking long-term dialogue context and enhance the performance of spoken language understanding by a large margin.

In addressing modeling the rich graph structures in dialogue utterances, we introduce a new Graph Convolutional LSTM (GC-LSTM) to learn the semantics contained in the graph-structured dialogues by incorporating a powerful graph convolutional operator. Our proposed GC-LSTM can not only capture the spatio-temporal semantic features in a dialogue, but also learn the co-occurrence relationship between intent detection and slot filling. Furthermore, we propose a Graph-to-Sequence learning framework to push the performance of spoken language understanding to a new start-of-the-art.

In addressing effectively integrating knowledge into dialogue generation systems, we propose a novel end-to-end learning framework to incorporate an external knowledge base (KB) and to capture the intrinsic graph semantics of the dialog history. We propose a novel Graph Memory Network (GMN) based sequence-to-sequence model, GraphMemDialogue, to effectively learn the inherent structural information hidden in dialog history, and to model the dynamic interaction between dialog history and KBs. We adopt a modified graph attention network to learn the rich structure representation in the dialog history, whereas the context-aware representation of KB entities are learnt by our novel GMN. To fully exploit this dynamic interaction, we design a learnable memory controller coupled with external KB entity memories to recurrently incorporate dialog history context into KB entities through a multi-hop reasoning mechanism. Experiments on three public datasets show that our GraphMemDialog model achieves state-of-the-art performance and outperforms strong baselines by a large margin, especially on datatests with more complicated KB information.

Chapter 1

Introduction

Dialogue systems, also known as conversational agents, are designed to communicate with a human via natural language. Typically, these dialogue systems range from chit-chatting or entertaining on topics like a movie star, to completing a specific task for example booking tickets, or reserving restaurants. Developing an intelligent dialogue system has been one of the longest running goals in artificial intelligence (AI). Recently, human computer conversation has attracted tremendous attention due to its promising commercial value, and been greatly advanced with the unprecedented progress of big data and AI. For example, they are widely deployed in personal assistants such as Apple Siri, Amazon Alexa, Microsoft XiaoIce, Google Assistant etc. On one hand, large amounts of conversational data easily accessible on the internet enables dialogue systems to learn how to talk like human beings. This greatly allows us to build data-driven, and open-domain conversation systems. On the other hand, the astounding breakthroughs in deep learning (DL) and reinforcement learning (RL) are practically applied to conversational AI in capturing complex patterns. Indeed, a large body of literatures has emerged to leverage a massive amount of data equipped with DL techniques to advance dialogue systems.

1.1 Motivation

Dialogue systems can be categorized into two classes based on their applications: chit-chat conversational systems and task-oriented dialogue systems. Chit-chat systems are designed to establish a general conversation with users. The main characteristics of chit-chat systems are open-domain and without any clear goal associated. Xiaoice from Microsoft is a typical example in industry. Task-oriented dialogue systems, on the other hand, aim to complete a specific task by interacting with users via dialogues, for example, booking restaurants. Representative task-oriented dialogue systems include Amazon Alexa, Google Home, etc. In both cases, the system requires to generate coherent responses to be aligned with dialogue history, and also needs to retrieve external knowledge like knowledge bases etc. Examples of chit-chat systems and task-oriented dialogue systems are shown in Table 1.1. In this thesis, we are primarily focusing on task-oriented dialogue systems.

	Chit-chat Dialogue Sytems	**Task-oriented Dialogue Sytems**
User	Hello, Xiaoice	Can I have coffee please?
Bot	Hello, your profile pic is interesting. Like I've seen it somewhere.	What coffee would you like?
User	Really?	I would like a cup of Mocha.
Bot	Is it a comic character?	Hot or Iced?
User	How do you know?	Iced with extra cream.
Bot	Guess	What size?
User	Are you human?	Grande.
Bot	Well, you will know it after chatting with me more.	You are all set.

Table 1.1: Dialogue examples for chit-chat and task-oriented dialogue systems.

Traditionally task-oriented dialog systems have been built modularly in a pipelined manner, namely designing each essential module individually, including spoken language understanding (SLU) (Chen et al., 2016), dialog state tracking (DST) (Zhong et al., 2018; Wu et al., 2019a), dialogue management (DM) (Young, 2006; Young et al., 2013a), and natural language generation (NLG) (Chen et al., 2019; Huang et al., 2020). SLU module parses user utterances into predefined semantic slots and intents. Then DST manages the output of SLU along with the dialogue history and maintains the dialogue states. The DM module subsequently takes dialogue states and produces dialogue actions for the next utterance. Finally NLG maps the selected action to its surface and generates final system responses.

Although these modularized systems are known to be stable and easy to interpret by combining domain-specific knowledge, they inherently expose a variety of major drawbacks. First, designing each component still requires heavily handcrafted rules and labels with domain-specific expert knowledge, especially in modules for dialogue state tracking and dialogue policy. For example, dialogue policy requires experts to label dialogue actions and slot information. This makes these modules highly difficult to be adapted to new domains. Second, modularized systems suffers intrinsically from the credit assignment problem (Zhao and Eskenazi, 2016), where the end uses' feedback is hard to be propagated to each upstream module. The last issue is process interdependence (Chen et al., 2017). The input of a component is dependent on the output of another one, for example in a pipelined manner. When adapting one component to a new domain or retraining it with new data, all the other components need to be adapted accordingly to ensure a global optimization. Slots and features might change accordingly. This process requires significant human effort.

Alternatively, with the advancement of deep learning techniques, end-to-end neural frameworks have become dominant to design task-oriented dialogue systems. These end-to-end neural models learn a distribution vector representation of the dialogue states automatically and directly map dialogue history to system responses. Generally an encoder-decoder model is applied to train the whole system in a supervised fashion. Thus end-to-end models make no assumption on the dialogue state structure and eliminate the dependency on human labels. Such property makes end-to-end neural models easily scalable to new domains and highly attractive. Furthermore, as recurrent neural networks (RNNs) emerge to represent the dialogue state using its latent memory and model the sequential dependencies in the dialogues, RNN-based end-to-end neural models have promptly become prevailing to build domain-agnostic dialog systems in both academia and industry. However, existing end-to-end methods in task-oriented dialogue systems still suffer from the following drawbacks: **1) Modeling of long-term slot context in SLU.** Though the latent memory of RNNs can model history information, they are inherently unstable over long time sequences because the memories are the RNN hidden states. (Weston et al., 2014) observes that RNNs tend to focus more on short-term memories and forcefully compress historical records into one hidden state vector. Thus, simple RNNs cannot preserve well long-term slot context of the conversation, which is crucial to future slot tagging in SLU. **2) Modeling the graph structure in dialogue utterances.** Dominant RNNs focus primarily on modeling sequential dependencies, and thus rich graph structure information hidden in the dialogue context is ignored. **3) Modeling graph-structured dialogue knowledge.** Effectively incorporating external knowledge into end-to-end task-oriented dialog systems still remains a challenge. It typically requires incorporating an external knowledge base (KB) and capturing the intrinsic semantics of the dialog history. Recent research shows promising results by using Sequence-to-Sequence models, Memory Networks, and even Graph Convolutional Networks. However, these mainstream approaches fail to effectively model context-aware and graph-structured dialogue knowledge.

1.2 Thesis Statement

In this dissertation, our major research objective is to model long-term dialogue context and to effectively incorporate dialogue knowledge information including dialogue history and external knowledge. The core research idea is to compensate mainstream RNN-based models's weakness aforementioned and to effectively learn graph-structured semantics in task-oriented dialogue systems.

Specifically, we first extend RNNs to memorize longer dialogue context by introducing a key-value memory networks for the sake of modeling long-term slot context in SLU. What's more, the semantic interaction between slots and intents is fully explored by utilizing the gated memory information to perform intent detection, mutually improving both tasks through global optimization. More importantly, we explore to optimize task-oriented dialogue systems from a new perspective, namely, modeling graph structures in dialogue systems. We propose a novel Graph Convolutional LSTM (GC-LSTM) encoder to learn the semantics contained in the graph-structured dialogues by incorporating a powerful graph convolutional operator. Our proposed GC-LSTM can not only capture the spatio-temporal semantic features in a dialogue, but also learn the co-occurrence relationship between intent detection and slot filling. Additionally, we propose a new Graph Memory Network (GMN) based sequence-to-sequence model, GraphMemDialogue, to effectively learn the inherent structural information hidden in dialog history, and to model the dynamic interaction between dialog history and KBs. We adopt a modified graph attention network to learn the rich structure representation in the dialog history, whereas the context-aware representation of KB entities are learnt by our novel GMN. To fully exploit this dynamic interaction, we design a learnable memory controller coupled with external KB entity memories to recurrently incorporate dialog history context into KB entities through a multi-hop reasoning mechanism. In short, we hope our exploration on learning graph structures in task-oriented dialogue systems could encourage brand-new methodologies to be developed and advance dialogue and natural language processing research to some extent.

1.3 Thesis Outline

In this dissertation, Chapter 3 is based on our paper titled "Spoken Language Understanding for Task-oriented Dialogue Systems with Augmented Memory Networks." Chapter 4 is based on our paper titled "A Graph-to-Sequence Model for Joint Intent Detection and Slot Filling in Task-Oriented Dialogue Systems." Chapter 5 is based on our paper "GraphMemDialog:

Optimizing End-to-End Task-Oriented Dialog Systems Using Graph Memory Networks." The rest of the thesis is organized as follows:

- **Chapter 2: Background and Related Work.** This chapter gives an overview of related research areas on task-oriented dialogue systems, sequence learning with recurrent networks, and graph convolutional networks.
- **Chapter 3: Spoken Language Understanding with Augmented Memory Networks.** This chapter presents a key-value memory network learning framework to effectively model long-term slot context in SLU. We further model the semantic interaction between slots and intents by utilizing the gated memory information to perform intent detection, mutually improving both tasks through global optimization.
- **Chapter 4: Graph-to-Sequence Learning Framework.** This chapter introduces a novel Graph-to-Sequence learning framework to model both temporal dependencies and structural information in a dialogue conversation. We introduce a new GC-LSTM encoder to learn the semantics contained in the graph-structured dialogues by incorporating a powerful graph convolutional operator. Our proposed GC-LSTM not only captures the spatio-temporal semantic features in a dialogue, but also learns the co-occurrence relationship between intent detection and slot filling.
- **Chapter 5: GraphMemDialogue: End-to-End Dialogue Modeling.** This chapter presents a novel GraphMemDialogue to effectively learn the inherent structural information hidden in dialog history, and to model the dynamic interaction between dialog history and KBs. We adopt a modified graph attention network to learn the rich structure representation in the dialog history, whereas the context-aware representation of KB entities are learnt by our novel GMN. To fully exploit this dynamic interaction, we design a learnable memory controller coupled with external KB entity memories to recurrently incorporate dialog history context into KB entities through a

multi-hop reasoning mechanism.

- **Chapter 6: Conclusion and Future Work.** This chapter concludes our main contributions and discusses possible future research directions.

Chapter 2

Background and Related Work

This chapter presents an overview of statistical spoken dialogue systems and their core components that pave the foundation for this dissertation. We first go over the background of existing models for building dialog systems and previous state-of-the-art approaches. Then we summarize fundamental deep learning techniques that this dissertation builds on.

2.1 Task-Oriented Dialogue Systems

Conventionally task-oriented dialogue systems were modularized as shown in Figure 2.1. The automatic speech recognition (ASR) module transcribes the user's speech input into natural language text format. The speech transcripts are then passed to the spoken language understanding (SLU) to decoding semantic frames, which typically involve performing domain identification, intent detection, and slot filling. The decoded information is then fed into the dialogue state tracking (DST) to maintain the current dialogue's states. The DST output is passed to the dialogue management (DM) module to generate the dialogue action by querying an external KB. Subsequently, natural language generation (NLG) generates the natural language response based on the previously produced dialogue action. Finally, the text to speech (TTS) module transforms the generated text into speech and sends it back to the users. In this thesis, we focus primarily on SLU and end-to-end NLG in task-oriented dialogue systems. Next we review each component in detail.

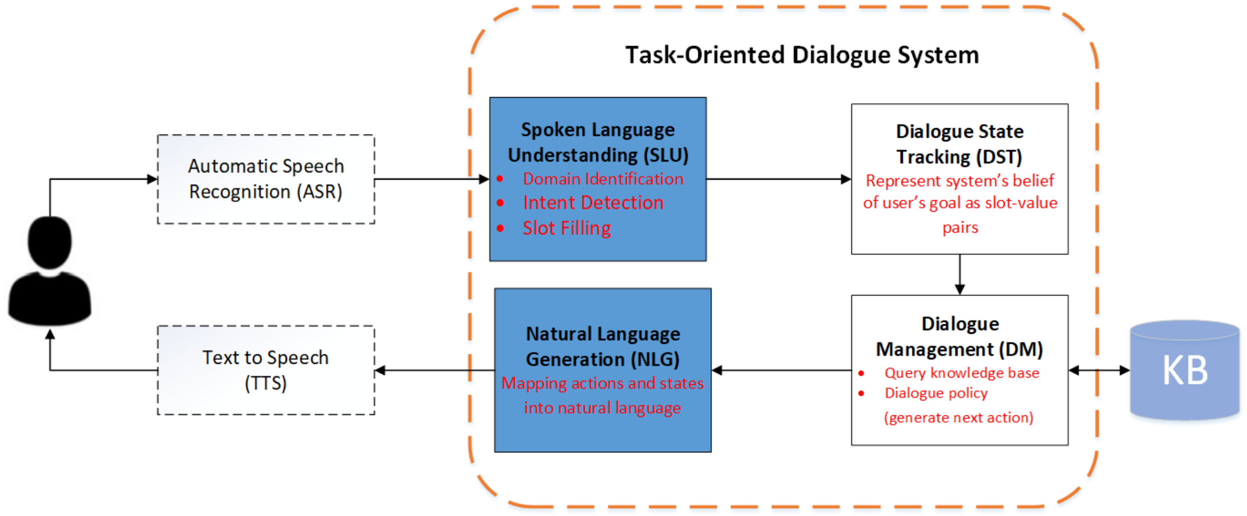
Task-Oriented Dialogue System
Automatic Speech Recognition (ASR)
Spoken Language Understanding (SLU)
Domain Identification
Intent Detection
Slot Filling
Dialogue State Tracking (DST)
Represent system's belief of user's goal as slot-value pairs
Text to Speech (TTS)
Natural Language Generation (NLG)
Mapping actions and states into natural language
Dialogue Management (DM)
Query knowledge base
Dialogue policy (generate next action)
KB

Figure 2.1: The major components of the modularized task-oriented dialogue systems.

2.1.1 Spoken Language Understanding

SLU is a key yet challenging component to parse users' utterances into semantic frames in order to capture a conversation's core meaning. It typically includes intention detection and slot filling. The former is to determine the user's intention, whereas the latter is to map each word in a utterance into predefined slot types. For example in Table 2.1, given an utterance "Flights from Charlotte to Miami", SLU is supposed to determine users' intention as *Flight* and to map each word in this utterance into predefined semantic slot types such as "B-fromloc" and "B-toloc".

Sentence	Flights	from	Charlotte	to	Miami
Intent	Flight				
Slots	O	O	B-fromloc	O	B-toloc

Table 2.1: An example utterance annotated with its intent and semantic slots (IOB format).

Intent Detection

Intent detection is formulated as an utterance classification problem and different classification methods. Support vector machines (SVM) and RNNs (Haffner et al., 2003; Sarikaya et al., 2011), have been proposed to solve it. Given a sequence of words $w = (w_1, w_2, ..., w_n)$, the goal of intent detection is to output an expected intent label o^I from a pre-defined set of intent classes for this whole sequence.

Slot Filling

Slot filling is treated as a sequence labeling task that maps the input utterance w into a predefined slot sequence $o^S = (o_1^S, o_2^S ..., o_n^S)$. Traditionally, hidden markov models (HMM) and conditional random fields (CRF) (Lee et al., 1992; Ye-Yi Wang et al., 2005; Raymond and Riccardi, 2007) were used to solve the slot filling problem. Later RNN based methods had become popular. For example, Yao et al. (2013) and Mesnil et al. (2015) employed RNNs for sequence labeling in order to perform slot filling.

Joint Intent Detection and Slot Filling

Early studies modeled intent detection and slot filling separately in a pipelined manner, and were insufficient to take full advantage of all supervised signal, as they intrinsically shared semantic knowledge. What's more, in the pipelined architecture, errors made in upper stream modules may propagate and be amplified in downstream components, which, however, could possibly be eased in joint models (Zhang and Wang, 2016). Thus recently, jointly modeling intent detection and slot filling has attracted significant attention, and achieved promising results with the aid of RNNs (Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018; Qin et al., 2019). The input of joint SLU is a sequence of words, whereas the output is a sequence of predefined slot types. A specific intent label is also assigned for the whole sentence.

Zhang and Wang (2016) first proposed joint work using RNNs for learning the correlation between intents and slots. Hakkani-Tür et al. (2016) adopted a RNN for slot filling and the last hidden state of the RNN was used to predict the utterance intent. Liu and Lane (2016) introduced an attention-based RNN encoder decoder model to jointly perform intent detection and slot filling. An attention weighted sum of all encoded hidden states was used to predict the utterance intent. All those models outperform the pipeline models via mutual enhancement between intent detection and slot filling.

Most recently, some work modeled the intent information for slot filling explicitly in the joint model. Goo et al. (2018) and Li et al. (2018) proposed the gate mechanism to explore incorporating the intent information for slot filling. However, as the sequence becomes longer, it is risky to simply rely on the gate function to sequentially summarize and compress all slots and context information in a single vector (Cheng et al., 2016). Wang et al. (2018) proposed the bi-model to consider the cross-impact between the intent and slots and achieve state-of-the-art results. Zhang et al. (2019) proposed a hierarchical capsule neural network to model the hierarchical relationship among words, slots, and intents in an utterance. Niu et al. (2019) introduced a SF-ID network to establish the interrelated mechanism for slot filling and intent detection tasks. However, these RNN-based models suffer from being weak at capturing long-range dependencies. Then Wu et al. (2021) explicitly modeled the long-term slot context knowledge via a key-value memory network beneficial to both slot filling and intent detection. Unfortunately all these models did not take the rich structure information in dialogues into consideration. Subsequently, Zhang et al. (2020) attempted to address the limitation of sequential models by utilizing S-LSTM with a context-gated mechanism to learn the local context in dialogue utterances, and achieved promising improvement compared with sequential RNN models.

Later in this thesis, we will discuss how we jointly model intent detection and slot filling to improve the overall performance from two different perspectives, that is, modeling long-term slot context and graph-structured semantics in the conversations.

2.1.2 Dialogue State Tracking

DST is a core component to ensure robustness in task-oriented dialogue systems and aims to estimate the user's goal at each dialogue turn. It also manages the input of each turn along with the dialogue history and outputs the current dialogue state. The dialogue state is usually represented in terms of a list of goal slots and the probability distribution of candidate values for each slot. Traditionally, DST was tackled with hand-crafted rules to select the most likely results (Goddeau et al., 1996). However, such systems are excessively dependent on human effort and suffer constantly from being weak at modeling uncertainties in ASR and SLU (Henderson, 2015). Thus, learning-based DSTs have been proposed to address these limitations. Henderson et al. (2013) presented a discriminative approach for tracking the state of a dialog which takes advantage of deep learning. It used a sliding window to output a sequence of probability distributions over an arbitrary number of possible values. Mrkšić et al. (2017) proposed a neural belief tracker model to reason over pre-trained word vectors, learning to compose them into distributed representations of user utterances and dialogue context, and matched the performance of state-of-the-art models relying on hand-crafted semantic lexicons.

2.1.3 Dialogue Management

DM learns the next action based on current dialogue states. A dialogue action usually consists of a speech action (e.g. request, inform) and slot-value pairs. Typically, a rule-based agent was introduced to warm-start the system (Yan et al., 2017). Then supervised learning was used on the actions generated by the rules. On the other hand, the dialogue policy can be trained end-to-end with reinforcement learning. Cuayáhuitl et al. (2015) applied deep reinforcement learning on strategic conversation that simultaneously learned the feature representation and dialogue policy.

2.1.4 Natural Language Generation

NLG maps the selected dialogue action to its surface and generates a response. Conventional approaches typically map input semantic symbols into intermediary representations of utterances such as tree-like or template structures, and then convert them into final response through surface realization (Walker et al., 2002). However, it is very costly to maintain templates and rules, thus leading to challenges in adapting to new domains. Furthermore, the quality of these NLG systems is limited by that of hand-crafted templates and rules. Recently, neural-based approaches have attracted significant attention in NLG. Wen et al. (2015b) presented a statistical language generator based on a semantically controlled Long Short-term Memory (LSTM) structure. The LSTM generator can learn from unaligned data by jointly optimising sentence planning and surface realisation using a simple cross entropy training criterion, and language variation can be easily achieved by sampling from output candidates. Wen et al. (2015a) employed a forward RNN generator, a CNN reranker, and backward RNN reranker to jointly generate utterances conditioned by the required dialogue act. Zhou et al. (2016) proposed a Context-Aware LSTM model to incorporate the question information, semantic slot values, and dialogue act type to generate correct answers. An attention mechanism was used to attends the key information in questions conditioned on the current decoding state of the decoder. Also encoding dialogue act type embedding further enabled the model to generate variant answers in response to different act types.

2.1.5 End-to-End Dialogue Systems

Modularized dialogue systems have been shown to be stable and provide the flexibility that allows each module to be designed independently. However, designing each component still requires heavily handcrafted rules and labels with domain-specific expert knowledge. This makes these modules highly difficult to be adapted to new domains. Furthermore, modularized dialogues lead to more complex design and improvement on individual modules does not translate into the whole system improvement. Finally, when adapting one component to a new domain or retraining it with new data, all the other components need to be adapted accordingly to ensure a global optimization. Slots and features might change accordingly. This process requires significant human effort.

In addressing these issues, fully data-driven end-to-end models are promising to build domain-agnostic dialogue systems based on recurrent neural networks (Zhao et al., 2017; Lei et al., 2018). Especially, end-to-end task-oriented dialogue systems (Serban et al., 2016; Wen et al., 2017) with sequence-to-sequence (seq-to-seq) models have attracted significant attention due to great flexibility and good quality. The core idea of a seq-to-seq model is to leverage an encoder to directly map the dialog history and KB to a vector representation, which is then fed into a decoder to generate a response word by word. Furthermore, Memory networks (MemNN) (Sukhbaatar et al., 2015) were employed to effectively incorporating dialog history context and knowledge bases (Madotto et al., 2018; Wu et al., 2019b). Multi-hop mechanisms further enabled MemNN to perform knowledge reasoning to select most relevant entities for generating a dialog response. Eric and Manning (2017a) proposed to use key-value memory networks to integrate a knowledge base by using a key memory to represent the subject and relation and a value memory to learn the object. Although memory network based models show promising results, MemNN suffers inherently from being weak at representing temporal dependencies between memories, which affects the conversational semantics due to ignoring the utterance order. On the other hand, pointer memories with copy mechanism proposed by Madotto et al. (2018) can effectively integrate the KB into dialog responses. Wu et al. (2019b) incorporated global pointer mechanism and achieved improved performance. Unfortunately, none of previous work has considered the rich structural information inherently in dialog history and the KB defined by entity-entity relations.

Subsequently, Graph Convolutional Networks (GCNs) have emerged as state-of-the-art methods for modeling knowledge graphs where entities are treated as nodes and their relations are edges. Specifically, Banerjee and Khapra (2019) proposed to use GCNs to model the word dependency associated with utterances, and entity relations in a knowledge base. Yang et al. (2020) proposed a new recurrent cell architecture to learn the dependency graph of the dialog history. Although those methods have modeled the structural information of the dialog history and KBs, they have ignored the strong correlation between them. Using dialog history as context should have a large impact on KB entity representations. He et al. (2020) was dedicated to modeling the association between the dialog history and KBs by using Flow operation (Huang et al., 2019) and Relational Graph Convolutional Networks (RGCN) (Schlichtkrull et al., 2017) to learn the KB entity representation. However, this work does not explicitly model the interaction between dialogue context and KBs, a key to make KB representation fully context-aware.

In our thesis, we propose a novel graph memory network (GMN) framework in learning context-aware KB entity graph representation. None of existing GCNs is leveraged to achieve this goal and we empirically show that our GMN outperforms some prevailing GCNs in modeling graphical KBs.

2.2 Sequence Learning with Recurrent Networks

Language is an inherently temporal phenomenon (Jurafsky and Martin, 2020). Recently, with the increasing popularity of neural models, there has been growing interests in learning this temporal nature via deep learning architectures. Recurrent neural networks offer a new way to represent the prior context, enabling the model's decision to depend on information from long distance words in the past.

2.2.1 Recurrent Neural Networks

A recurrent neural network as shown in Figure 2.2 is any network that contains a cycle within its network connections, meaning that the value of some unit is directly, or indirectly dependent on its own earlier outputs as an input (Jurafsky and Martin, 2020). This allows it to exhibit temporal dynamic behavior. Derived from feed-forward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs (Hyötyniemi, 1996). A widely used RNN architecture is referred as **Elman Networks** (Elman, 1990), or simple recurrent networks. The Elman RNN feeds the hidden layer output at time $t-1$ back to the same hidden layer at time t via recurrent connections. Thus, information stored in the hidden layer can be viewed as a summary of input sequence up till the current time. A non-linear and differentiable activation function is applied to the weighted sum of the input vector and previous hidden state. The hidden state at time t can thus be formulated as:

$$h(t) = \sigma\left(Ux(t) + Wh(t-1)\right) \tag{2.1}$$

where σ is an activation function. As feed-forward networks, a training set, a loss function, and back-propagation are used to adjust the weights in the recurrent networks. Specifically, network parameters are optimized based on loss functions derived using maximum likelihood, and updated using back-propagation through time method considering influence of past states through recurrent connections. Error from the output layer is back-propagated to the hidden layers through the recurrent connections backwards in time. The weight matrices are updated after each training sample.

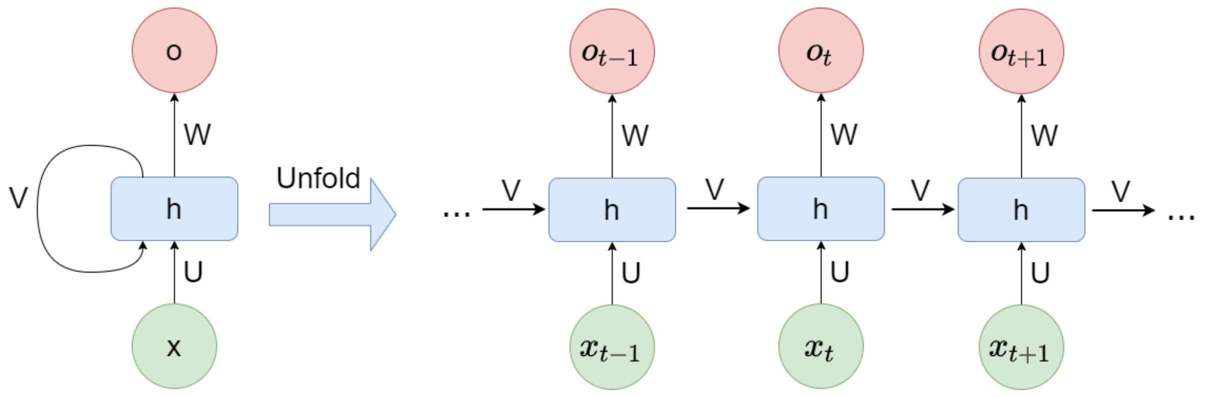

Figure 2.2: A diagram of a RNN. Compressed version on the left, unfold version on the right.

In practice, it is quite difficult to train RNNs on long sequences due to vanishing/exploding gradients problems. The vanishing and exploding gradient phenomena are often encountered in the context of RNNs. The reason why they happen is that it is difficult to capture long term dependencies because of multiplicative gradient that can be exponentially decreasing/increasing with respect to the number of layers. To address these issues, more complex network architectures have been proposed, such as, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Units (GRUs) (Cho et al., 2014b). Both are variances of original RNNs to track long-term dependencies while mitigating the vanishing/exploding gradients problems. A GRU has a reset gate and an update gate. The reset gate is used from the model to decide how much of the past information to forget, whereas the update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future. On the other hand, The

LSTM has three gates, namely, input gate, forget gate, and output gate, thus having more parameters than GRUs. In this thesis, we use both LSTM and GRUs for different scenarios. But since LSTM can be seen as a generalization of GRU, we only go over LSTM in detail for simplicity.

2.2.2 Long Short-Term Memory

LSTM models replace the recurrent model that uses sigmoid or hyperbolic tangent activation function with the memory block. The memory block may contain one or more memory cells. A common LSTM unit is composed of a cell, an input gate, an forget gate and a output gate. The cell remembers summarized information of previous observations, whereas three gates regulate the flow of information into and out of the cell. Specifically, the input gate i_t regulates how much of the new cell state to keep, the forget gate f_t regulates how much of the existing memory to forget, and the output gate o_t regulates how much of the cell state should be exposed to the next layers of the network. Mathematically, A LSTM can be formulated as follows:

$$\begin{aligned}
f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) \\
i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i) \\
\tilde{C}_t &= \tanh(W_c h_{t-1} + U_c x_t + b_C) \\
C_t &= f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \\
o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) \\
h_t &= o_t \odot \tanh(C_t)
\end{aligned} \tag{2.2}$$

where σ is the sigmoid function, and $\odot$ denotes Hadamard product.

These controlling gates in memory block learn to forget information that is no longer needed and to remember information required for future decisions. This mechanism effectively avoids gradient exploding and vanishing problems.

2.2.3 Encoder-Decoder Models

Encoder-decoder networks, or seq-to-seq networks are a special class of Recurrent Neural Network architectures used to solve complex language problems like Machine Translation, Question Answering, Chatbots, and Text Summarization. In natural language processing, encoder-decoder networks remain a powerful conditional generative model (Cho et al., 2014a; Serban et al., 2017). An encoder-decoder network as shown in Figure 2.3 maps a fixed-length input to a fixed-length output where the length of the input and output may differ. It models the target word sequence Y conditioned on given input word sequence X, that is, $P(Y|X)$. Generally, an encoder-decoder network consists of three major components:

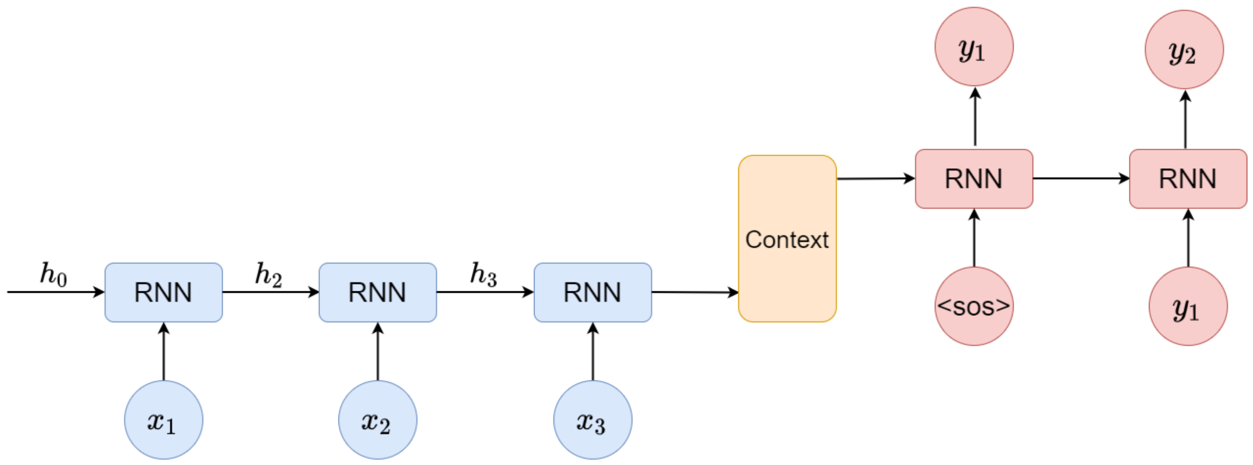

Figure 2.3: The encoder-decoder architecture.

- **Encoder**. An encoder accepts an input sequence $\{x_1 ... x_n\}$ and generates a sequence of contextualized representations $\{h_1^E ... h_n^E\}$. Words in vocabulary are usually represented by word embeddings, which are either randomly initialized and learned during training, or pre-trained such as ELMo (Peters et al., 2018), GloVe (Pennington et al., 2014), etc. Formally, the input sequence is encoded by recursively applying:

$$h_i^E = \text{RNN}^E(\phi^{emb}(x_i), h_{i-1}^E) \tag{2.3}$$

where $h_0^E = 0$.

- **Context Vector**. A context vector c is a function of h_i^E, conveying the essence of the input of the decoder. One simple way is to treat the last hidden state h_n^E as the representation of the input sequence, that is:

$$c = h_n^E \tag{2.4}$$

- **Decoder**. A decoder takes the context vector c as input and generates decoding hidden state h_i^D. Finally the decoder predicts the target sequence words one-by-one conditioned on all previously generated tokens as follows:

$$h_j^D = \mathrm{RNN}^D(\phi^{emb}(y_{j-1}), h_{j-1}^D) \tag{2.5}$$

$$o_j = \mathrm{Softmax}(W h_j^{dec} + b) \tag{2.6}$$

where o_j is the output probability of every word in the vocabulary at each decoding step j, y_{j-1} is the previous emitted word, and W and b are trainable parameters. In order to make the model to predict the first word and to end the prediction, special tokens SOS (start-to-sentence) and EOS (end-of-sentence) tokens are padded at the beginning and the end of the target sequence. The SOS token is important for the decoder. Because the decoder progresses by taking the tokens it emits as inputs along with the embedding and hidden state, it needs some kind of tokens to start with before it has emitted anything. SOS token fits this scenario.

2.3 Graph Convolutional Networks

Graph convolutional networks (GCNs) generalize convolutional neural networks to graphs and are an effective framework for learning the representation of graph structured data. There are two types of GCNs: spatial GCNs and spectral GCNs. In spatial GCNs, a convolution operation is applied to compute a new feature vector for each node using its neighborhood information. For example, Simonovsky and Komodakis (2017) formulated a convolution-like operation on graph signals performed in the spatial domain and applied graph convolutions to point cloud classification. On the other hand, spectral GCNs transform graph signals on graph spectral domains and then apply spectral filtering on graph signals. Defferrard et al. (2016) proposed a spectral formulation for the convolutional operator on graph with fast localized convolutions. Kipf and Welling (2017) introduced Spectral GCNs for semi-supervised classification on graph-structured data.

2.3.1 GCNs for Undirected Graphs

In this section, we describe GCNs (Kipf and Welling, 2017) for undirected graphs. GCNs compute representations for the nodes of the graph by aggregating information from their neighborhood nodes. We can stack l layers of GCNs to account for neighbors that are l-hops away from the current node. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ be an undirected graph, where $\mathcal{V}$ is the set of nodes ($|\mathcal{V}| = n$), $\mathcal{E}$ is the set of edges, and $\mathcal{A} \in \mathbb{R}^{n \times n}$ is an adjacent matrix. Let $\mathcal{X} \in \mathbb{R}^{n \times m}$ be the input feature matrix with n nodes and each node is represented by a m-dimensional feature vector. The hidden representation $H^{(l+1)} \in \mathbb{R}^{n \times d}$ with d-dimension at $(l+1)$-th layer is defined as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right) \tag{2.7}$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. I_N is the identity matrix, and $\tilde{D}^{ii} = \sum_j \tilde{A}_{ij}$. $W^{(l)}$ is a trainable weight matrix in l-th layer, and σ is an activation function. $H^{(0)}$ is set to $\mathcal{X}$.

2.3.2 Graph Convolution LSTM

In order to incorporate temporal features, Seo et al. (2016) firstly proposed a graph convolutional LSTM to capture the spatial-temporal features of graph structures. This was an extension of GCNs to have the recurrent architecture. Cui et al. (2019) introduced a Traffic Graph Convolutional LSTM to learn the interactions between roadways in the traffic networks. Si et al. (2019) further improved graph convolutional LSTM network by introducing attention to effectively extract discriminative spatial and temporal features in Skeleton-Based Action Recognition.

However, none of these Graph Convolutional LSTM models dialogue graphs in single turn conversational systems. In this thesis, inspired by Seo et al. (2016) and Zhang et al. (2020), we propose a novel GC-LSTM to effectively learn the spatial and temporal representation on dialogue utterances by combing graph convolution with S-LSTM (Zhang et al., 2018), not the conventional LSTM. To the best of our knowledge, our work is the first one that introduces a Graph-to-Seq learning framework with Graph Convolution for joint intent detection and slot filling in task-oriented dialogue systems.

Chapter 3

Spoken Language Understanding with Augmented Memory Networks

In addressing the weakness of RNNs processing long sequences, in this chapter, we propose a RNN model with augmented memory networks to track long-term slot context. We first present the architecture of our neural network based model in detail. Then we discuss system learning, system evaluation and performance results.

Spoken language understanding, usually including intent detection and slot filling, is a core component to build a spoken dialog system. Recent research shows promising results by jointly learning of those two tasks based on the fact that slot filling and intent detection are sharing semantic knowledge. Furthermore, the attention mechanism boosts joint learning to achieve state-of-the-art results. However, current joint learning models ignore the following important facts: **1.** Long-term slot context is not traced effectively, which is crucial for future slot filling. **2.** Slot tagging and intent detection could be mutually rewarding, but bi-directional interaction between slot filling and intent detection remains seldom explored. In this chapter, we propose a novel approach to model long-term slot context and to fully utilize the semantic correlation between slots and intents. We adopt a key-value memory network to model slot context dynamically and to track more important slot tags decoded before, which are then fed into our decoder for slot tagging. Furthermore, gated memory information is utilized to perform intent detection, mutually improving both tasks through global optimization. Experiments on benchmark ATIS and Snips datasets show that our model achieves state-of-the-art performance and outperforms other methods, especially for the slot filling task.

3.1 Introduction

Task-oriented dialogue systems have attracted significant attention, which have been greatly advanced by deep learning techniques. SLU, including intention detection and slot filling (Tur and Mori, 2011), is a key yet challenging problem to parse users' utterances into semantic frames in order to capture a conversation's core meaning. Traditionally, intention detection is treated as a classification problem, whereas slot filling is usually defined as sequence labeling problem, where In-Out-Begin (IOB) format is applied for representing slot tags. Given an utterance, SLU determines users' intention and maps it into predefined semantic slots. The input is a sequence of words, and the output is a sequence of predefined slot IDs. A specific intent is assigned for the whole sentence.

In the traditional pipeline approach, intent detection and slot filling are implemented separately. However, separate modeling of those two tasks is insufficient to take full advantage of all supervised signals, as they share semantic knowledge. For example, if the intent of an utterance is "find_a_flight", it is more likely to contain slots "departure_city" and "arrival_city" rather than "restaurant_name". Another drawback of the pipeline method is that errors made in upper stream modules may propagate and be amplified in downstream components, which however could possibly be eased in joint model (Zhang and Wang, 2016).

Recently, joint model for intent detection and slot filling has been proposed and achieved promising results (Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018). Though achieving promising performance, their models suffer from two major issues: **1) Modeling of slot context.** Though the latent memory of RNNs can model history information, they are inherently unstable over long time sequences because the memories are the RNN hidden states. (Weston et al., 2014) observes that RNNs tend to focus more on short-term memories and forcefully compress historical records into one hidden state vector. Thus, simple RNNs cannot preserve long-term slot context of the conversation, which is crucial to future slot tagging. **2) Bi-directional interaction between slot filling and intent detection.** The majority of joint modeling work has studied how to utilize intent information to improve slot filling performance. However, the beneficial impact of slot information on intent detection is mostly ignored. In fact, slots and intents are closely correlative, thus mutually reinforcing each other.

In this chapter, we propose a new framework to jointly model intent detection and slot filling in order to achieve a deeper level of semantic modeling. Specifically, our model is distinguished from previous work primarily in two ways.

- **Model slot context dynamically with Key-Value Memory Networks (KV-MNs).** The majority of existing work use RNNs to track slot values mentioned in previous utterances. However, RNNs tend to focus more on short-term memories. We propose to use a memory network to model slot context information as external knowledge which is acting a global information to guide slot tagging. Instead of relying on the compressed vector in RNN, KV-MNs store different historical slot tag information separately in different memory slots, which enriches the representation capacity compared with RNNs. Furthermore, slot values mentioned in the utterance are dynamically tracked, which is beneficial for subsequent slot tagging at each timestamp. Lastly, slot-level attention can model more accurately the contribution of each word in an utterance to slot tagging.

- **Model the mutual interaction between intent detection and slot filling.** The fact that intent detection and slot filling are semantically related is well-observed and how to use intent information to boost slot filling is widely explored. However, slot filling is beneficial to intent detection as well, and these benefits are yet to be explored. We propose a gating mechanism between intents and slots based on KV-MNs in order to model the interaction between intent detection and slot filling.

In the following, we demonstrate how memory networks can be used to model long-term slot context knowledge and the interaction between intent detection and slot filling in single turn conversational systems.

3.2 Proposed Model

Memory networks show promising results on learning long-range dependency, but they are insensitive to represent temporal dependencies between memories (Wu et al., 2018). RNNs tend to be opposite. Thus, it makes sense for us to combine those networks together to model long-term slot context information. In this section, we present a specific key-value dynamic memory module to collect and remember slot clues in the dialog context. Then context memory is used to enhance the Encoder-Decoder based model to perform slot filling and intent detection.

As illustrated in Figure 3.1, our proposed model is composed of an Encoder-Decoder, and a Key-Value Memory Module including KEY-MEMORY, VALUE-MEMORY, a memory read unit, and a memory write unit. Given a single-turn dialog, the Encoder transforms a word in user utterances into a dense vector by using a shared self-attentive encoder. Then the memory network encodes long-term slot context information by incorporating historical slot

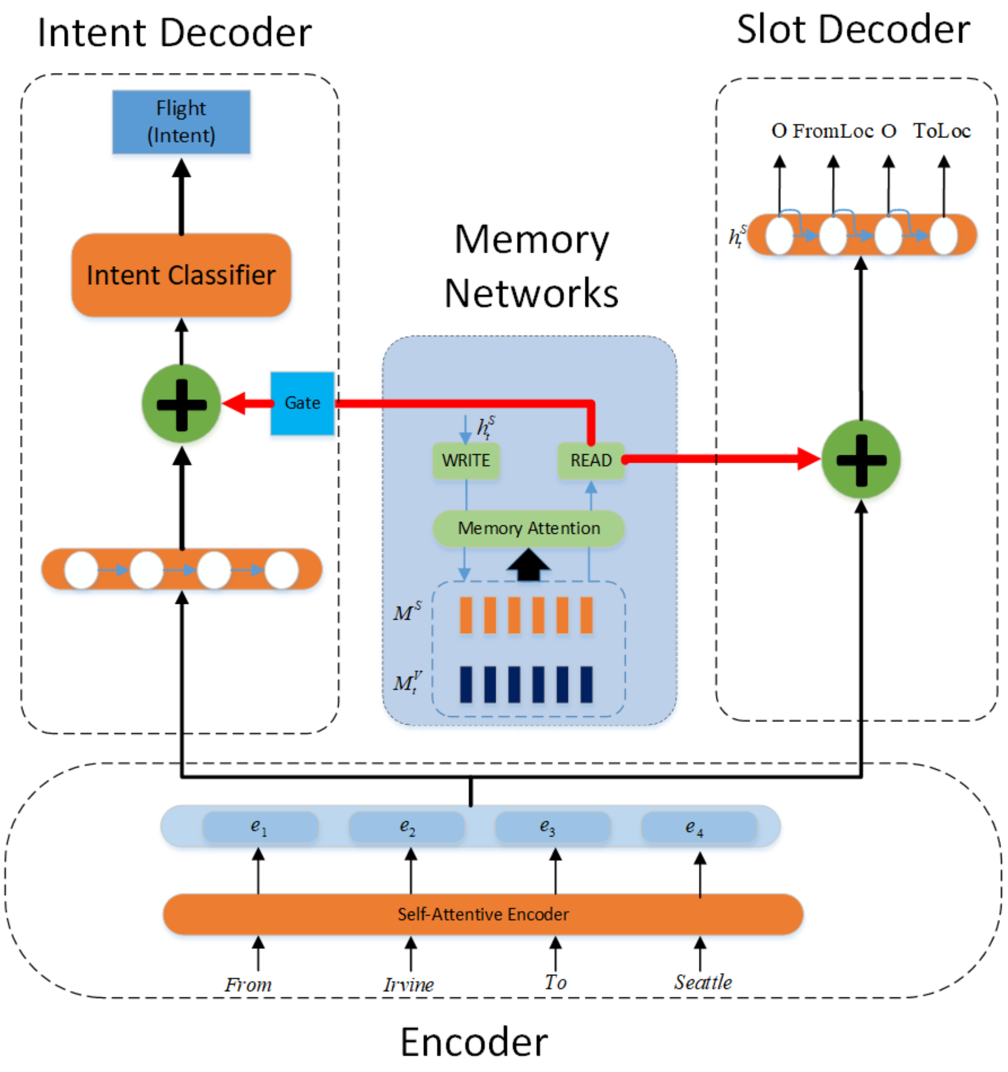

Figure 3.1: Framework of the proposed model

tags through memory attention and WRITE operations of the memory network. The slot decoder integrates short-term hidden state of self-attention encoder and the long-term slot context generated by attentively reading the VALUE-MEMORY to generate slot tagging at each timestamp. Later, intent decoder performs token level intent detection, which is seen as a coarse-grained intent detection result. Finally, a fine-grained intent detection is produced by gating memory modules. Both intent detection and slot filling are optimized simultaneously via a joint learning scheme.

3.2.1 Self-Attentive Encoder

Given an input utterance $\mathbf{X} = (x_1, x_2, \ldots, x_T)$ of T words, where each word is initially represented by a vector of dimension d, the BiLSTM (Hochreiter and Schmidhuber, 1997) is applied to learn representations of each word by reading the input utterance forward and backward to produce context sensitive hidden states $\mathbf{H} = (h_1, h_2, \ldots, h_T)$:

$$h_t = \mathbf{BiLSTM}(x_t, h_{t-1}) \tag{3.1}$$

Then, we use self-attention mechanism to capture the contextual information for each token. We adopt the method proposed by (Vaswani et al., 2017a), where we first map the matrix of input vectors $\mathbf{X} \in \mathbb{R}^{T \times d}$ to queries ($\mathbf{Q}$), keys ($\tilde{\mathbf{K}}$) and values ($\tilde{\mathbf{V}}$) matrices by using different linear projections and the self-attention output $\mathbf{C} \in \mathbb{R}^{T \times d_1}$ is:

$$\mathbf{C} = \text{softmax}\left(\frac{\mathbf{Q}\tilde{\mathbf{K}}^\top}{\sqrt{d_2}}\right)\tilde{\mathbf{V}} \tag{3.2}$$

where d_1 and d_2 represents self-attention dimension and keys'dimension. We concatenate the output of self-attention and BiLSTM as the final encoding representation as shown in Qin et al. (2019):

$$\mathbf{E} = \mathbf{H} \oplus \mathbf{C} \tag{3.3}$$

where $\mathbf{E} = (e_1, \ldots, e_T) \in \mathbb{R}^{T \times (d+d_1)}$ and $\oplus$ is a concatenation operation.

3.2.2 Slot Decoder

Our slot deocder consists of two components: **1)** the key-value memory-augmented attention model which generates slot context representation of users' utterance, and **2)** the unidirectional LSTM decoder, which predicts the next slot tag step by step.

Dynamic Key Value Memory Network

To overcome the shortcomings of RNNs in capturing semantic clues over the long-term, we design a memory network that can preserve fine-grained semantic information of long-term slot context. We adopt a key-value memory network, which memorizes information by using a large array of external memory slots. The external memories enrich the representation capability compared with hidden vectors of RNNs and enable the KV-MNs to capture long-term data characteristics (Liu and Perez, 2017). We incorporate the knowledge contained in the historical slot tags into the memory slots. The KV-MNs decompose slot semantics in an utterance into different slot categories and thus preserves more fine-grained information. In KV-MNs, a memory slot is represented by a key vector and an associated value vector.

- **KEY-MEMORY:** The KEY-MEMORY $\mathbf{K} \in \mathbb{R}^{d_k \times n}$ learns latent correlation between utterance words and slot tags, where n is the number of memory slots and d_k is the dimension of each slot. Each column vector, that is, i-th key vector $k_i \in \mathbb{R}^{d_k}$ is set to the i-th column of the KEY-MEMORY $\mathbf{K}$, which is shared by all conversation turns and fixed during the processing of word sequences.

- **VALUE-MEMORY:** Both the KEY-MEMORY and VALUE-MEMORY have the same number of memory slots. Each value memory vector stores the value of slot tag mentioned in the utterance. We form a value memory matrix $\mathbf{V}_t \in \mathbb{R}^{d_v \times n}$ by combining all n value slots. Different from KEY-MEMORY $\mathbf{K}$, VALUE-MEMORY $\mathbf{V}_t$ is word-specific and is continuously updated according to the input word sequence. During the conversation, the value of a new slot tag may be added into the VALUE-MEMORY, and an old value can be erased. In this way, we can adequately capture the slot context information on each mentioned slot. Two types of operations, **READ** and **WRITE**, are designed to manipulate the value memories.

Memory-augmented Decoder

As shown in Figure 3.1, the decoder uses the aligned BiLSTM hidden state h_t as a query to address the KEY-MEMORY looking for an attention vector a_t, and attentively reads the VALUE-MEMORY to generate slot context representation c_t.

First, we use h_t to address the KEY-MEMORY to find an accurate attention vector a_t.

$$a_t = \mathbf{Address}\left(h_t, \mathbf{K}\right) \tag{3.4}$$

a_t is subsequently used as the guidance for reading the VALUE-MEMORY $\mathbf{V}_{t-1}$ to get the slot context representation c_t.

$$c_t = \mathbf{Read}(a_t, \mathbf{V}_{t-1}) \tag{3.5}$$

c_t works together with the aligned encoder hidden state e_t to generate the new decoder state at the decoding step t,

$$h_t^S = \mathbf{LSTM}\left(h_{t-1}^S, y_{t-1}^S, e_t \oplus c_t\right) \tag{3.6}$$

where h_{t-1}^S is the previous slot decoder state and y_{t-1}^S is the previous emitted slot lable distribution. After that, we use the slot decoder hidden state h_t^S to update $\mathbf{V}_t$:

$$\mathbf{V}_t = \mathbf{Write}\left(h_t^S, \mathbf{V}_{t-1}\right) \tag{3.7}$$

Finally, the decoder state h_t^S is utilized for slot filling:

$$y_t^S = \text{softmax}\left(\mathbf{W}_h^S h_t^S\right) \tag{3.8}$$

$$o_t^S = \text{argmax}\left(y_t^S\right) \tag{3.9}$$

where $\mathbf{W}_h^S$ are trainable parameters and o_t^S is the slot label of the word at timestamp t in the utterance.

3.2.3 Intent Detection Decoder

Different than most existing work where intent information is used to do slot filling, our framework is directly leveraging the explicit slot context information to help intent detection. Furthermore, a gated mechanism is used in order to effectively incorporate slot memory information into intent detection. By performing gated intent detection, there are two advantages:

1. Sharing slot context information with intent detection improves intent detection performance since those two tasks are related. Furthermore, a gating mechanism which combines the intent detection information and slot context retrieved from key-value memory, regulates the degree of enhancement of intent detection to prevent information overload.
2. Through shared key-value memory, the interaction between intent detection and slot filling can be effectively modeled and executed. Plus, by jointly training those two tasks, not only can intent detection performance be improved by slot context knowledge, but also slot filling is enhanced by minimizing intent detection objective function. In other words, by learning optimal parameters of shared key-value memory, slot filling and intent detection interact in a more effective and deeper way.

Intent Detection Decoder

For intent detection, we use another uni-directional LSTM as the intent detection network. At each decode step t, the decoder state h_t^I is generated by the previous decoder state h_{t-1}^I, the previous emitted intent label distribution y^I_{t-1} and the aligned encoder hidden e_t.

$$h^I_t = \textbf{LSTM}\left(h^I_{t-1}, y^I_{t-1}, e_t\right) \tag{3.10}$$

Then the intent decoder state h^I_t together with the slot context c_t is utilized for final intent detection.

Gated Memory

We propose a gated mechanism to integrate slot context with intent detection. The gate regulates the degree of slot context information to feed into the intent detection task and prevent information from overloading. As shown in Figure 3.2, the gate **G** is a trainable fully connected network with sigmoid activation.

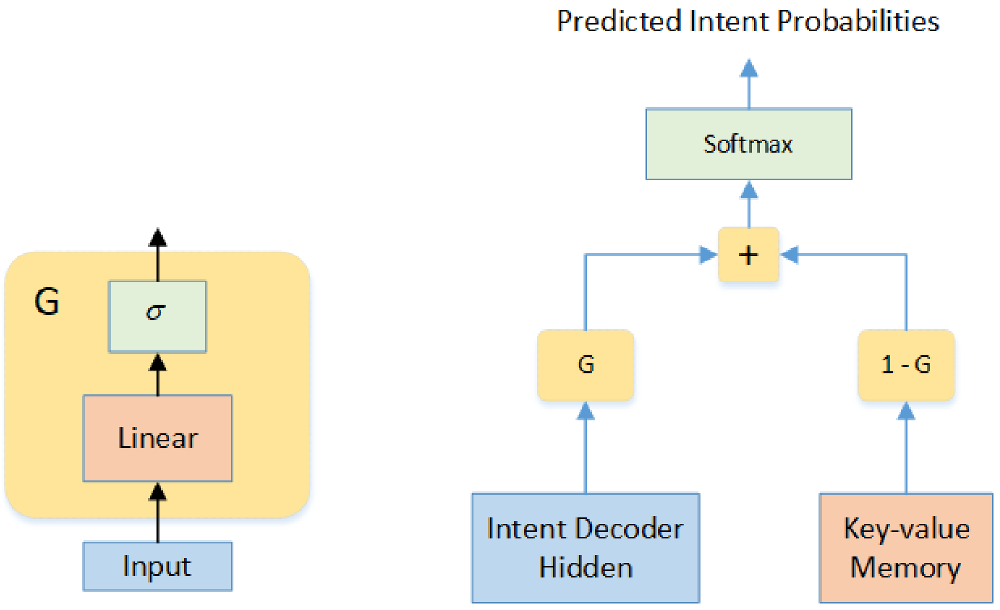

Figure 3.2: Intent detection with gated memory

$$h'^I_t = g_t \cdot h^I_t + (1 - g_t) \cdot c_t \tag{3.11}$$

where $g_t = \text{sigmoid}\left(W_t[h_t^I \bigoplus c_t] + b_t\right)$. Then, the output of gated decoder state h'^I_t is utilized for intent detection:

$$y_t^I = \text{softmax}\left(W_h^I h'^I_t\right) \tag{3.12}$$

$$o_t^I = \text{argmax}(y_t^I) \tag{3.13}$$

where y_t^I is the intent output distribution of the t-th token in the utterance, o_t^I represents the intent lable of t-th token and W_h^I are trainable parameters of the model.

The final utterance result o^I is generated by voting from all token intent results as illustrated in Qin et al. (2019).

3.2.4 Memory Access Operation

In this section, we detail how to access key-value memory at the decoding time step t.

KEY-MEMORY Address: $\mathbf{K} \in \mathbb{R}^{d_k \times n}$ denotes the KEY-MEMORY at decoding time step t. The addressed attention vector is given by

$$a_t = \textbf{Address}\left(h_t, \mathbf{K}\right) \tag{3.14}$$

where $a_t \in \mathbb{R}^n$ specifies the normalized weights assigned to the slots in $\mathbf{K}$, with j-th slot being k_j. The attention weights $a_{t,j}$ are calculated based on the correlation between h_t and k_j:

$$a_{t,j} = \frac{exp(e_{t,j})}{\sum_{i=1}^{n} exp(e_{t,i})} \tag{3.15}$$

where $e_{t,j} = k_j^\top \left(\mathbf{W}_a h_t + b_a\right)$

VALUE-MEMORY Read: $\mathbf{V}_t \in \mathbb{R}^{d_v \times n}$ denotes the VALUE-MEMORY at decoding time step t. The output of reading the value memory $\mathbf{V}_t$ is given by

$$c_t = \sum_{j=1}^{n} a_{t,j} \mathrm{v}_{t,j} \tag{3.16}$$

VALUE-MEMORY Write: Similar to the attentive writing operation of neural turing machines (Graves et al., 2014), we define two types of operation for updating the VALUE-MEMORY: FORGET and ADD.

FORGET determines the content to be removed from memory slots. More specifically, the vector $\mathbf{F}_t \in \mathbb{R}^{d_v}$ specifies the values to be forgotten or removed on each dimension in memory slots, which is then assigned to each memory slot through normalized weights a_t. We use the slot decoder hidden state h_t^S to update $\mathbf{V}_{t-1}$. Formally, the memory after FORGET operation is given by

$$\tilde{\mathrm{v}}_{t,i} = \mathrm{v}_{t-1,i}(\mathbf{1} - a_{t,i} \cdot \mathbf{F}_t), i = 1, 2, \ldots, n \tag{3.17}$$

where

- $\mathbf{F}_t = \sigma(\mathbf{W}_F, h_t^S)$ is parameterized with $\mathbf{W}_F \in \mathbb{R}^{d_v \times d_h}$, and δ stands for the *Sigmoid* activation function, and $\mathbf{F}_t \in \mathbb{R}^{d_v}$;
- $a_t \in \mathbb{R}^n$ specifies the normalized weights assigned to the key memory slots in $\mathbf{K}$, and $a_{t,i}$ represents the weight associated with the i-th memory slot.

ADD decides how much current information should be written to the memory as the added content:

$$\mathrm{v}_{t,i} = \tilde{\mathrm{v}}_{t,i} + a_{t,i} \cdot \mathbf{A}_t, i = 1, 2, \ldots, n \tag{3.18}$$

where $\mathbf{A}_t = \sigma(\mathbf{W}_A, h_t^S)$ is parameterized with $\mathbf{W}_A \in \mathbb{R}^{d_v \times d_h}$ and $\mathbf{A}_t \in \mathbb{R}^{d_v}$. By learning the parameters of FORGET and ADD layers, our model can automatically determine which signal to weaken or strengthen based on input utterance words.

3.2.5 Joint Training

The loss function for intent detection is $\mathcal{L}_1$, and that for slot filling is $\mathcal{L}_2$, which are defined as cross entropy:

$$\mathcal{L}_1 \triangleq -\sum_{j=1}^{m}\sum_{i=1}^{n_I} \hat{y}_j^{I,i} \log\left(y_j^{I,i}\right) \tag{3.19}$$

and

$$\mathcal{L}_2 \triangleq -\sum_{j=1}^{m}\sum_{i=1}^{n_S} \hat{y}_j^{S,i} \log\left(y_j^{S,i}\right) \tag{3.20}$$

where $\hat{y}_j^{I,i}$ and $\hat{y}_j^{S,i}$ are the gold intent label and gold slot label respectively, m is the number of words in a word sequence, and n_I and n_S are the number of intent label types and the number of slot tag types, respectively.

Finally the joint objective is formulated as weighted-sum of these two loss functions using hyper-parameters α and β:

$$\mathcal{L}_\theta = \alpha\mathcal{L}_1 + \beta\mathcal{L}_2 \tag{3.21}$$

Through joint training, the key-value memory shared by those two tasks can learn the shared representations and interactions between them, thus further promoting each other's performance and easing the error propagation compared with pipeline models.

3.3 Experimental Setup

3.3.1 Datasets

To evaluate our proposed model, we conduct experiments on two widely used benchmark datasets: ATIS (Airline Travel Information System) and Snips. Both datesets used in our experiements follow the same format and partition as in Goo et al. (2018).

ATIS. This dataset (Hemphill et al., 1990) contains audio recordings of people making flight reservations. The training set has 4,478 utterances and the test set contains 893 utterances. We use another 500 utterances for the development set. There are 120 slot labels and 21 intent types in the training sets.

Snips. To justify the generalization of our proposed mode, we also execute our experiment on another NLU dataset collected by Snips (Coucke et al., 2018) [1]. This data is collected from the Snips personal voice assistant, where the number of samples for each intent is approximately the same. The training set contains 13,804 utterances and the test set contains 700 utterances. We use another 700 utterances as the development set. There are 72 slot labels and 7 intent types. Compared to single-domain ATIS dataset, Snips is more complicated mainly due to its large vocabulary, and to the diversity of intents and slots (Goo et al., 2018). For example, *GetWeather* and *BookRestaurant* in Snips are from different topics, resulting in a larger vocabulary. On the other hand, intents in ATIS are all about flight information with similar vocabularies.

[1] `https://github.com/snipsco/nlu-benchmark/tree/master/2017-06-custom-intent-engines`

3.3.2 Training Details

We implement our model in Pytorch, which is trained on NVIDIA GeForce GTX 1080 Ti. In our experiments, we set the dimension of word embedding to 256 for ATIS and 200 for Snips dataset. L2 regularization used in our model is 1×10^{-6} and dropout ratio is set to 0.4 for reducing overfit. The number of memory columns is set to 20 for both datasets, and the dimensions of memory column vectors are set to 64 for ATIS, and to 200 for Snips. The optimizer is Adam (Kingma and Ba, 2014). During our experiments, we select the model which works the best on the development set, and then evaluate it on the test set.

3.3.3 Automatic Evaluation Metrics

In order to have fair comparison with others' work, we adopt three most popular evaluation metrics in SLU studies (Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018; Niu et al., 2019).

- **Intent Accuracy.** This is the classification accuracy. Some utterance in the ATIS dataset has more than one intent labels. Following(Goo et al., 2018), we require all of these intents have to be correctly predicted in order to be counted as a correct classification.

- **F1 Score.** We assess slot filling performance by using the F1 score. It is defined as the harmonic average of the precision and recall. Precision measures the percentage of the items that system detected that are in fact positive. Precision is defined as:

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{3.22}$$

  Recall measure the percentage of items actually present in the input that were correctly identified by the system. Recall is defined as:

$$\text{Recall} = \frac{\text{true\_positives}}{\text{true\_positives} + \text{false\_negatives}} \tag{3.23}$$

Hence, F1 is defined as:

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3.24}$$

- **Sentence (Sent.) Accuracy.** A sentence prediction is counted as correct unless all of its slot labels and intent are accurately predicted. This metric takes both slot filling and intent detection into consideration.

3.3.4 Baselines

We compare our model with the following baselines:

- Joint Seq. (Hakkani-Tür et al., 2016). This model proposed a RNN-LSTM architecture to jointly model domain detection, intent detection, and slot filling on multi-domains.

- Attention BiRNN (Liu and Lane, 2016). An attention-based neural network model for joint intent detection and slot filling was proposed to learn the relationship between slots and intents.

- Sloted-Gated (Goo et al., 2018). A gate mechanism was introduced to explore incorporating the intent information for slot filling in a joint model.

- CAPSULE-NLU (Zhang et al., 2019). This model proposed a capsule-based neural network model which accomplishes slot filling and intent detection via a dynamic routing-by-agreement schema. A rerouting schema was proposed to further synergize the slot filling performance using the inferred intent representation.

- SF-ID Network (Niu et al., 2019). This model introduced a SF-ID network to establish the interrelated mechanism for slot filling and intent detection tasks.

- Stack-Propagation (Qin et al., 2019). This model adopted a joint model with Stack-Propagation which could directly use the intent information as input for slot filling. In addition, the token-level intent detection was performed to further alleviate the error propagation.

When doing the comparison, we adopt the reported results from those papers directly.

3.4 Experimental Results and Analysis

3.4.1 Experimental Results

Table 3.1 shows the experiment results of the proposed model on ATIS and Snips datasets. From the table, we can see that our model outperforms all the baselines in all three aspects: slot filling (F1), intent detection (Acc) and setence accurancy (Acc), demonstrating that explicitly modeling slot context and strong relationships between slots and intent can benefit SLU effectively from the key-value memory. In the ATIS dataset, compared with the best prior joint work Stack-Propagation (Qin et al., 2019), we achieve F1 score as 96.13 which is even slightly better than Stack-propagation's F1 score (96.10) with BERT model. This signifies that our key-value memory can not only capture long-term slot context, but also model correlation between slot filling and intent detection, which can be further optimized by joint training. What's more, in the Snips dataset, our model achieves good results in both slot filling and overall sentence. Specifically, slot filling was improved by almost 1.0%, and sentence accuracy by 1.4%. Generally, ATIS dataset is a simpler SLU task than Snips, and so the room to be improved is relatively small. On the other hand, Snips is more complex so that it needs more complicated model to capture long-term context and share the knowledge across different topics.

Model	ATIS Dataset			Snips Dataset		
	Slot(F1)	**Intent(Acc)**	**Sent.(Acc)**	**Slot(F1)**	**Intent(Acc)**	**Sent.(Acc)**
Joint Seq.(Hakkani-Tür et al., 2016)	94.30	92.60	80.70	87.30	96.90	73.20
Attention BiRNN(Liu and Lane, 2016)	94.20	91.10	78.90	87.80	96.70	74.10
Sloted-Gated(Goo et al., 2018)	95.42	95.41	83.73	89.27	96.86	76.43
CAPSULE-NLU(Zhang et al., 2019)	95.20	95.0	83.40	91.80	97.30	80.90
SF-ID Network(Niu et al., 2019)	95.58	96.58	86.00	90.46	97.0	78.37
Stack-Propagation(Qin et al., 2019)	95.90	96.90	86.50	94.20	98.0	86.90
Our model	**96.13**	**97.20**	**87.12**	**95.13**	**98.14**	**88.14**

Table 3.1: SLU Performance comparison on ATIS and Snips datasets (%). The improved results are written in bold.

3.4.2 Analysis

From Section 3.4.1, we can see good improvements on both datasets, but we want to know how each component impacts SLU performance.

Ablation Study

In this section, we explore how each component contributes to our full model. Specifically, we ablate three important scenarios and conduct them in this experiment. Note that all the variants are based on joint learning.

- Without key-value memory and gating architecture for integrating slot context information with intent detection. This is the model similar to Qin et al. (2019).
- Only with key-value memory, but without sharing slot context information with intent detection.
- With key-value memory and sharing, but without gating architecture, where only key-value memory is applied to model slot context and that information is directly fed into intent detection.

Model	ATIS Dataset			Snips Dataset		
	Slot(F1)	**Intent(Acc)**	**Sent.(Acc)**	**Slot(F1)**	**Intent(Acc)**	**Sent.(Acc)**
Without K-V memory and sharing	95.72	96.64	85.78	94.08	97.42	86.42
With K-V memory without sharing with intent	95.95	96.66	86.56	94.46	98.09	87.0
With K-V memory and sharing without gate	96.08	96.86	87.0	94.76	98.0	87.28
Full Model	**96.13**	**97.20**	**87.12**	**95.13**	**98.14**	**88.14**

Table 3.2: Feature ablation study on our proposed model on ATIS and Snips datasets (%)

Table 3.2 shows the joint learning performance of our model on ATIS and Snips datasets by removing one component at one time. First, if we remove key-value memory and gating architecture, the performance drops dramatically compared with our proposed model. This is expected as it does not have any of our improvements. Then we only consider key-value memory to model slot context. From Table 3.2, we can see that key-value memory does improve performance in a large scale. The result can be interpreted as indicating that key-value memory learns long-term slot context representation effectively, which does compensate the weakness of RNN. In the following, we apply key-value memory and also share it with intent detection without gating. It is noticeable that SLU performance is enhanced further. Sharing slot context information with intent detection not only improves intent accuracy, but also improves slot filling through joint optimization. Finally, when we add the gating mechanism, the performance improves further. We attribute this to the gating mechanism that regulates the degree of slot context information to feed into intent detection task and prevent information from overloading.

We also study how the number of memory slots and the dimension of memory slots impacts SLU performance. Figure 3.3 shows the performance change with different hyper-parameters. We found that the optimal size of memory slots for ATIS and Snips dataset is 20, whereas the optimal dimension of memory slots is 64 for ATIS and 200 for Snips respectively.

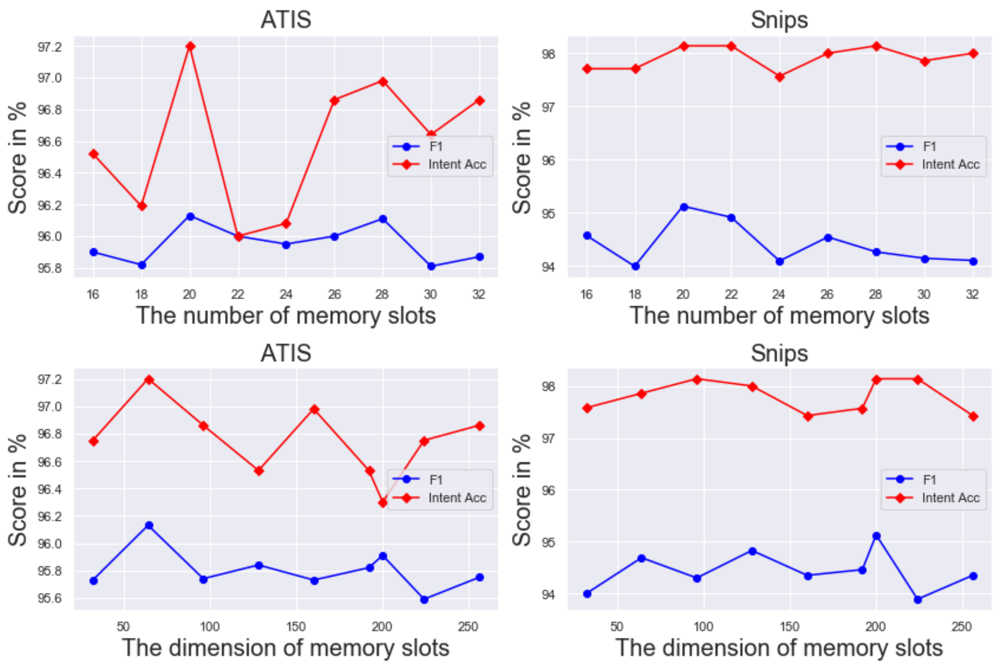

Figure 3.3: SLU performance on different hyper-parameters in key-value memory networks

Memory Attention

Analyzing the attention weights has been frequently used to show the memory read-out, since it is an intuitive way to understand the model dynamics. Figure 3.4 shows the attention vector for each decoded slot, where each row represents attention vector a_t. Our model has a sharp distribution over the memory, which implies that it is able to select the most related memory slots from the value memory. For example, when decoding "san", our model selects memory slot 1, 7, 8,15 from the value memory to read context information, where memory slot 7 and 15 are representing word "from" and memory slot 1 representing word "flight". In other words, words "flight" and "from" contribute more than other previous words in order to decode "san" to B-fromloc.city_name.

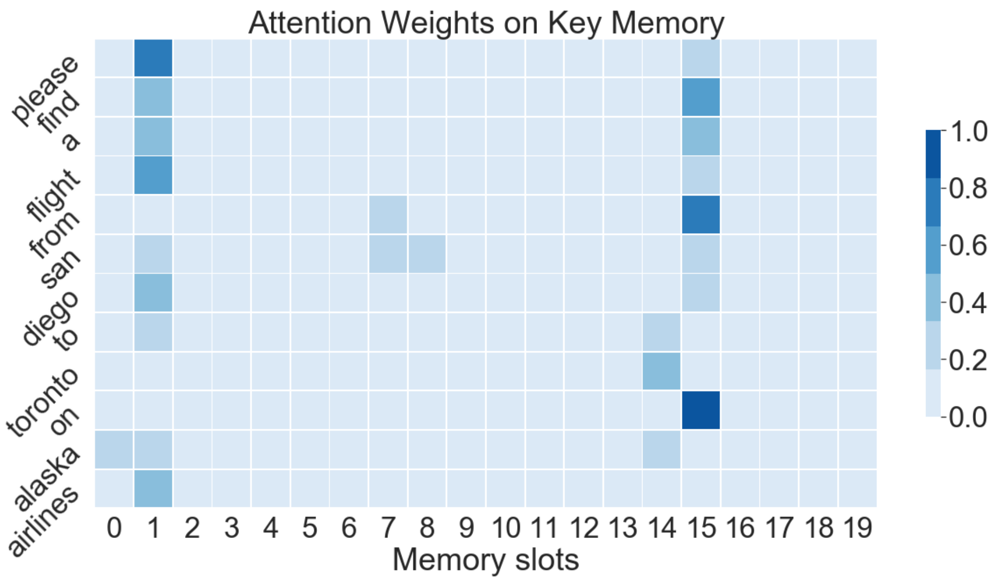

Figure 3.4: Key memory attention visualization from the ATIS dataset

3.5 Conclusions

In this chapter, we propose a joint model to perform spoken language understanding with an augmented key-value memory to model slot context in order to capture long-term slot information. In addition, we adopt a gating mechanism to incorporate slot context information for intent classification to improve intent detection performance. Reciprocally, joint optimization promotes slot filling performance further by memory sharing between those two tasks. Experiments on two public datasets show the effectiveness of our proposed model and achieve state-of-the-art results.

Chapter 4

Graph-to-Sequence Learning Framework

In this chapter, in addressing modeling the rich graph structures in dialogue utterances, we propose a novel Graph-to-Sequence learning framework to learn the semantics contained in the graph-structured dialogues.

Although RNN-based neural models show promising results by jointly learning of these two tasks, dominant RNNs are primarily focusing on modeling sequential dependencies. Rich graph structure information hidden in the dialogue context is seldomly explored. In this chapter, we propose a novel Graph-to-Sequence model to tackle the spoken language understanding problem by modeling both temporal dependencies and structural information in a conversation. We introduce a new Graph Convolutional LSTM (GC-LSTM) encoder to learn the semantics contained in the dialogue dependency graph by incorporating a powerful graph convolutional operator. Our proposed GC-LSTM can not only capture the spatio-temporal semantic features in a dialogue, but also learn the co-occurrence relationship between intent detection and slot filling. Furthermore, a LSTM decoder is utilized to perform final decoding of both slot filling and intent detection, which mutually improves both tasks through global optimization. Experiments on benchmark ATIS and Snips datasets show that our model achieves state-of-the-art performance and outperforms existing models.

4.1 Introduction

Although jointly modeling intent detection and slot filling has attracted significant attention, and achieved promising results with recurrent neural networks (RNNs) (Liu and Lane, 2016; Goo et al., 2018; Li et al., 2018). However, these RNN-based models are primarily focusing on modeling sequential dependencies, and inherently unstable over long-time sequences as RNNs tend to focus more on short-term memories (Weston et al., 2014). Indeed, this weakness of sequential RNN-based models leads to a large portion of slot filling errors (Tur et al., 2010).

Ignoring the graph structure information flowing along the conversations puts a bottleneck on state-of-the-art approaches. Subsequently, Zhang et al. (2020) attempted to address the limitation of sequential models by utilizing S-LSTM to learn the graph structure in dialogue utterances, and achieved promising improvement compared with sequential RNNs. Nevertheless, this model still suffers from three major issues: **1) Modeling dialogue graphs**. Although the n-gram context graph used in S-LSTM has to some extent captured the influence of neighboring words within a specific window, closely-related words, such as "parents", "children" and "siblings" in a dialogue graph can be outside this window and unfortunately neglected. These words should have substantial impact on slot tag decoding. Furthermore, unrelated words within the n-gram window are acting as noise, leading to more slot filling errors. **2) Learning spatial structures in dialogues**. S-LSTM is incapable of capturing spatial structures in a conversational context, and we observe that this property plays a vital role in modeling a fully graph-structured dialogue. **3) Jointly decoding intent detection and slot filling in a stand-alone decoder**. Zhang et al. (2020) utilized a S-LSTM to both encode dialogue states and decode final intents and slot tags. This puts too much burden on the S-LSTM and deteriorates SLU performance. In fact, sequence-to-sequence framework shows great advantages in sequence labeling, which we can leverage to free us from this dilemma.

In this chapter, we propose a novel Graph-to-Sequence (Graph-to-Seq) framework to perform joint intent detection and slot filling in task-oriented dialogue systems. The proposed model learns spatio-temporal semantic features hidden in dialogues by modeling dialogue structure as a dependency graph, and by employing a Graph Convolutional LSTM (GC-LSTM). Graph Convolutional operation further enables a deeper level of semantic modeling of the dialogue context. A separate SLU decoder is also used to jointly decode slot tags and intents in a globally optimal way.

In short, our contributions are threefold:

- To the best of our knowledge, our work is the first one that introduce spectral graph convolution to model the graph structures in SLU.
- We propose a novel GC-LSTM to effectively learn graph-structured representations in dialogues. We model a dialogue graph as an enhanced dependency tree by adding forward and backward edges between words in order to capture both sequential and structural information in dialogues.
- We introduce a novel Graph-to-Seq framework to perform joint SLU. A stand-alone RNN decoder is greatly beneficial to improve SLU performance by relieving encoders from decoding burden, and by enabling the interaction between intent detection and slot filling.

4.2 Graph-to-Sequence

Given a sequence of words $w = (w_1, w_2, ..., w_n)$ in an utterance and an associated dialogue dependency relation graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, intent detection can be defined as a classification task that outputs an expected intent label o^I, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of word nodes and relations in $\mathcal{G}$, and n is the utterance length. Slot filling can be seen as a sequence labeling task that maps the input utterance w into a predefined slot sequence $o^S = (o_1^S, o_2^S ..., o_n^S)$.

4.2.1 Model Overview

We propose a novel Graph-to-Seq framework to combine the merits of S-LSTM and GCNs to effectively learn the spatio-temporal representation of the dialogue context. Our proposed model is composed of two major components: a GC-LSTM encoder, and a SLU decoder, as shown in Figure 4.1. The GC-LSTM encoder learns fixed-length vectors to represent the dialogue context structurally. Not only can it model spatial graph structure information, but also it learns the semantic correlation between slots and intents. Message passing in our graph convolutional operation improves on capturing the long-range dependencies. We further add a context gate to improve our model's ability to utilize local context information. In our decoder, a dedicated LSTM is employed to integrate hidden states of the GC-LSTM for generating slot tagging and final intents. Our decoder first decodes slot tags and then outputs an expected intent. We intentionally choose this mechanism to improve intent accuracy since the slot information is beneficial to intent detection. Both intent detection and slot filling are optimized simultaneously via joint learning. In the following sections we detail each component thoroughly.

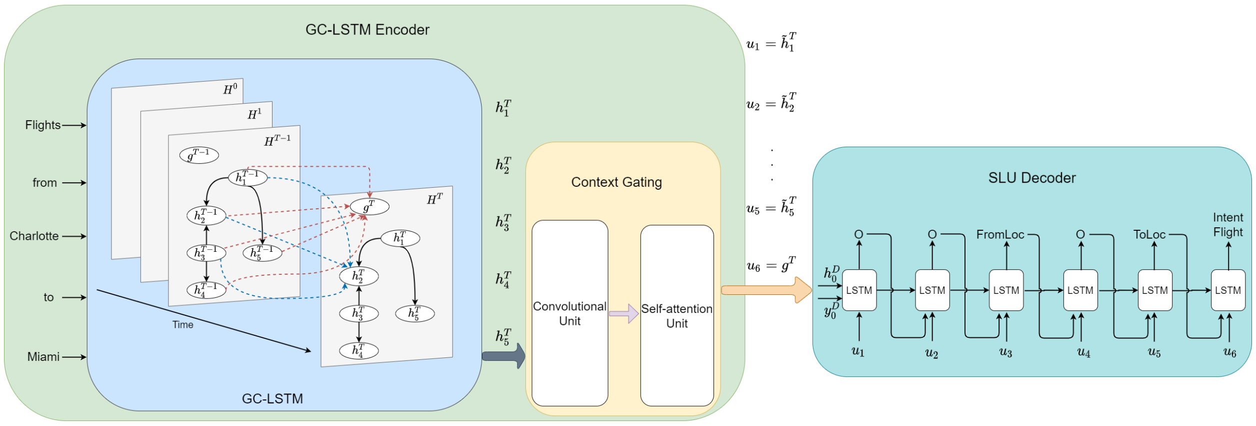

Figure 4.1: Graph-to-Sequence model for joint intent detection and slot filling.

4.2.2 Spectral Graph Convolutions

Graph convolutional neural networks are an effective framework for learning the representation of graph structured data. As it is difficult to express a meaningful translation operator in the vertex domain, Defferrard et al. (2016) proposed a spectral formulation for the convolutional operator on graph $*_{\mathcal{G}}$. Based on this definition, a spectral convolution on graphs is defined as the multiplication of a graph signal $x \in \mathbb{R}^N$ (a scalar for every node) with a non-parametric kernel filter $g_\theta = \operatorname{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in Fourier domain, where N is the number of vertices, as follows:

$$g_\theta *_{\mathcal{G}} x = U g_\theta U^T x \tag{4.1}$$

where $U \in \mathbb{R}^{N \times N}$ is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = U \Lambda U^T$ with a diagonal matrix of its eigenvalues Λ and $U^T x$ being the graph Fourier transform of x. g_θ can be thought as a function of eigenvalues of L, i.e. $g_\theta(\Lambda)$. However, evaluating Equation (4.1) is computationally expensive as the multiplication with U is $\mathcal{O}(N^2)$. Calculating the eigendecomposition of L might be prohibitively expensive for large graphs. Thus, Defferrard et al. (2016) parameterized g_θ as a truncated expansion, up to $(K-1)^{th}$ order of Chebyshev polynomials T_k such that:

$$g_\theta(\Lambda) \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \tag{4.2}$$

where the parameter $\theta \in \mathbb{R}^K$ is the vector of Chebyshev coefficients and $T_k(\tilde{\Lambda})$ is the Chebyshev polynomial of order k evaluated at $\tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_N$. λ_{max} denotes the largest eigenvalue of L. The graph filtering operation can then be written as

$$g_\theta *_\mathcal{G} x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x \tag{4.3}$$

where $\tilde{L} = 2L/\lambda_{max} - I_N$. Equation (4.3) can be evaluated by using the stable recurrent relation $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$ in $\mathcal{O}(K|\mathcal{E}|)$ operations, where $\mathcal{E}$ is the number of edges. As pointed out by Defferrard et al. (2016), when the filtering operation Equation (4.3) is an order K polynomial of the Laplacian, it is K-localized and depends only on nodes that are maximum K hops away from the central node, that is, the K-neighborhood.

4.2.3 Graph Convolutional LSTM Encoder

Based on the graph convolutional operation defined in Equation (4.3), we propose a GC-LSTM encoder to encode graph structures in dialogue utterances. Utterance words are first transformed to word embeddings $e = (e_1, e_2, ..., e_n)$ by using the pretrained ELMo embeddings (Peters et al., 2018). They are then fed into the GC-LSTM at each time step. After T steps, our GC-LSTM generates word-level hidden states for decoding slot labels, and sentence-level hidden states for predicting intents.

Dialogue Graph

We model word relationships by using dependency trees, as dependency links are close to the semantic relationships for the next stage of interpretation. In order to enable learning various relationships of words such as dependency relations, we first use the off-the-shelf parsing tool called Spacy[1] to extract dependency relation graph $\mathcal{G}$ among words in dialog utterances as shown in Figure 4.2. To further support bi-directional information flow, we explicitly add reverse edges and sequential relations (i.e., Next and Prev) as well. Enhanced dependency graphs allow information flow between dependent words and head words bidirectionally, enabling the learning process to capture the rich semantic representation between them.

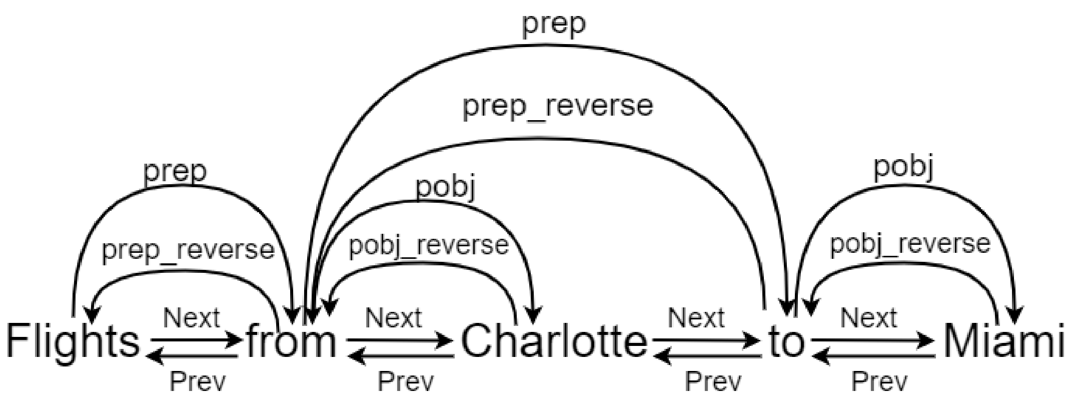

Figure 4.2: An example of our dialogue utterance graph

Graph Convolutional LSTM

We introduce a new GC-LSTM by extending S-LSTM (Zhang et al., 2018) to include a powerful graph convolutional operator in order to better learn graph-structured semantics in a dialogue. GC-LSTM views the whole sentence as a single graph $\mathcal{G}$, consisting of word-level nodes h_i and a sentence-level node g. At each time step t, the graph state is represented as: $H^t = (h_1^t, h_2^t, ..., h_n^t, g_t)$.

[1] https://spacy.io/

Like S-LSTM, GC-LSTM uses a recurrent state transition process to model information between sub states, which enriches state representations incrementally. Unlike S-LSTM, GC-LSTM updates its word hidden states using the graph convolution operation in order to capture spatial features of the semantics. The graph state transition from H^{t-1} to H^t consists of word-level node state change from h^{t-1} to h^t, and sentence-level state transition from g^{t-1} to g^t. We set initial state $h_i^0 = g^0 = h^0$ in H^0, where h^0 is a parameter. The hidden state h_i^t is a function of word embedding e_i, its neighboring node hidden state h_j^{t-1}, and sentence-level state g^{t-1}, where $j \in \mathcal{N}(i)$ and $\mathcal{N}(i)$ is the neighbor nodes of word node i. This updating function is formulated as follows:

$$\begin{aligned}
\hat{i}^t &= \delta(W_i *_{\mathcal{G}} h^{t-1} + U_i e^t + V_i g^{t-1} + b_i) \\
\hat{f}^t &= \delta(W_f *_{\mathcal{G}} h^{t-1} + U_f e^t + V_f g^{t-1} + b_f) \\
\hat{s}^t &= \delta(W_s *_{\mathcal{G}} h^{t-1} + U_s e^t + V_s g^{t-1} + b_s) \\
o^t &= \delta(W_o *_{\mathcal{G}} h^{t-1} + U_o e^t + V_o g^{t-1} + b_o) \\
u^t &= \delta(W_u *_{\mathcal{G}} h^{t-1} + U_u e^t + V_u g^{t-1} + b_u) \\
i^t, f^t, s^t &= \mathrm{softmax}(\hat{i}^t, \hat{f}^t, \hat{s}^t) \\
c^t &= f^t \odot c^{t-1} + s^t \odot c_g^{t-1} + i^t \odot u^t \\
h^t &= o^t \odot \tanh(c^t)
\end{aligned} \tag{4.4}$$

where W_x, U_x, V_x and b_x are model parameters ($x \in \{i, f, s, o, u\}$), δ is the sigmoid function, and $\odot$ denotes Hadamard product.

Different than S-LSTM, GC-LSTM only contains three gates: an input gate i^t, a forget gate f^t, and an output gate o^t. But similar to S-LSTM, GC-LSTM also has one sentence gate s_t controlling information from sentence cell c_g^{t-1}. These control gates are normalized by a softmax function and then serve as probability weights to regulate new cell states c_t and hidden states h^t.

Following Zhang et al. (2018), the sentence-level node g^t is updated based on all word-level nodes:

$$
\begin{aligned}
&\bar{h} = \text{avg}(h_1^{t-1}, h_2^{t-1}, ..., h_n^{t-1}) \\
&\tilde{f}_g^t = \delta(W_g g^{t-1} + U_g \bar{h} + b_g) \\
&\tilde{f}_i^t = \delta(W_f g^{t-1} + U_f h_i^{t-1} + b_{\tilde{f}}) \\
&\tilde{o}_g^t = \delta(W_{og} g^{t-1} + U_{og} \bar{h} + b_{og}) \\
&f'^t_1, ..., f'^t_n, f'^t_g = \text{softmax}(\tilde{f}_1^t, ..., \tilde{f}_n^t, \tilde{f}_g^t) \\
&c_g^t = f'^t_g \odot c_g^{t-1} + \sum_{i=1}^{n} f'^t_i \odot c_i^{t-1} \\
&g^t = o_g^t \odot \tanh(c_g^t)
\end{aligned}
\tag{4.5}
$$

where W_x, U_x and b_x are model parameters ($x \in \{g, f, og\}$). $f'^t_1, ..., f'^t_n$ and f'^t_g are gates controlling information from $c_1^{t-1}, ..., c_n^{t-1}$ and c_g^{t-1}, whereas o_g^t is an output gate regulating the recurrent cell c_g^t to g_t.

At each time step, word-level nodes capture an increasingly larger scope of the dialogue graph, building up knowledge incrementally. The sentence-level node gathers information from all the word-level nodes to refine the whole utterance representation. Slot nodes and the intent node are interacting with each other via Equations (4.4) and (4.5), which learns the correlation between intent detection and slot filling. Unlike an LSTM which uses only one state to represent the utterances sequentially, our GC-LSTM uses multiple states (i.e. n word-level states and 1 sentence-level state) to learn deeper context information incrementally with the aid of our graph convolutional operation. In this way, our GC-LSTM can capture long-range dependencies. Finally, after T time steps, word-level hidden states h^T and the sentence-level hidden state g^T are used for predicting slot labels and an expected intent.

Gated Context

In order to make our encoder fully context-aware, we introduce a context gate to capture the contextual information for each token as Zhang et al. (2020) did. The context gate includes a convolution unit and a self-attention unit. The convolution unit is supposed to capture local context information, such as the internal correlation of phrase structure. Following Zhang et al. (2020), we have multiple convolutional filters with variable sizes to capture different local context, i.e., creating three different representations as $conv^1$, $conv^2$, $conv^3$. Finally, convolved slot hidden states are obtained below:

$$\begin{aligned} z_i &= [conv_i^1 || conv_i^2 || conv_i^3] W_z \\ conv_i &= \text{ReLU}\left(W_c [h_{i-\frac{k}{2}}^T, ..., h_{i-\frac{k}{2}}^T] + b_c \right) \end{aligned} \tag{4.6}$$

where W_z is the model parameter, $||$ is a concatenation operation, and $conv_i$ is a 1-d convolution unit with W_c and b_c being model parameters, and k being the filter size.

Furthermore, we use a multi-head self-attention mechanism (Vaswani et al., 2017b) to capture diverse global contextual information. The self attention unit then takes input the output of the convolution unit to generate gated context hidden state $\tilde{z}$:

$$\tilde{z} = \text{MultiHeadAttn}(z, z, z) \tag{4.7}$$

where MultiHeadAttn is defined in Vaswani et al. (2017b).

Encoder Output

The final output of our GC-LSTM encoder with context gating is:

$$\begin{aligned} U^T &= (u_1^T, ...u_{n+1}^T) = [\tilde{h}_1^T, ..., \tilde{h}_n^T || g^T] \\ \tilde{h}_i^T &= h_i^T \odot \delta(\tilde{z}_i) \end{aligned} \tag{4.8}$$

4.2.4 SLU Decoder

Different than most existing joint models where intent detection and slot filling are decoded separately, our framework decodes them in a shared LSTM. We directly leverage the explicit slot decoding context information to help intent detection. By performing a joint SLU decoding in a stand-alone LSTM, there are two advantages:

1. The architecture for example in Zhang et al. (2020) puts too much burden on one Graph LSTM encoder as it is playing a dual role in both encoding and decoding. We observe that separating decoding from encoding can potentially free an encoder from this dilemma, thus beneficial to overall performance improvement. Indeed dominant seq-to-seq models are primarily relying on an independent autoregressive decoder to generate slot tokens one by one conditioned on all previously generated tokens.

2. Sharing slot decoding context with intent detection improves intent detection performance since those two tasks are related. This can be further substantiated by our experiment results. Through shared decoding states, the interaction between intent detection and slot filling can be effectively modeled and executed. In addition, by jointly training those two tasks, not only can intent detection performance be improved by slot context knowledge, but also slot filling is enhanced by minimizing intent detection objective function.

We use one unidirectional LSTM as a SLU decoder. At the decoding step $i \in [1, n+1]$, the decoder state h_i^D can be formalized as:

$$h_i^D = \text{LSTM}(h_{i-1}^D, y_{i-1}^D, u_i) \tag{4.9}$$

where $h_0^D = \tanh(W_0^D u_n)$, h_{i-1}^D is the previous decoder state, y_{i-1}^D is the previous emitted slot label distribution, and W_0^D is the model parameter.

Finally, the slot filling is given by:

$$y_i^S = \text{softmax}(W_h^S h_i^D) \quad i \in [1, n] \tag{4.10}$$

$$o_i^S = \text{argmax}(y_i^S) \tag{4.11}$$

The intent detection is defined as follows:

$$y^I = \text{softmax}(W_h^I h_{n+1}^D) \tag{4.12}$$

$$o^I = \text{argmax}(y^I) \tag{4.13}$$

4.2.5 Joint Training

The loss function for intent detection is $\mathcal{L}_1$, and that for slot filling is $\mathcal{L}_2$, which are defined as cross entropy:

$$\mathcal{L}_1 \triangleq -\sum_{i=1}^{n_I} \hat{y}^{I,i} \log\left(y^{I,i}\right) \tag{4.14}$$

and

$$\mathcal{L}_2 \triangleq -\sum_{j=1}^{n}\sum_{i=1}^{n_S} \hat{y}_j^{S,i} \log\left(y_j^{S,i}\right) \tag{4.15}$$

where $\hat{y}^{I,i}$ and $\hat{y}_j^{S,i}$ are the gold intent label and gold slot label respectively, and n_I and n_S are the number of intent label types and the number of slot tag types, respectively.

Finally the joint objective is formulated as follows by using hyper-parameters α:

$$\mathcal{L}_\theta = \alpha\mathcal{L}_1 + \mathcal{L}_2 \tag{4.16}$$

4.3 Experimental Setup

4.3.1 Datasets

To evaluate our proposed model, we conduct experiments on two widely used benchmark datasets: ATIS (Airline Travel Information System) and Snips as illustrated in Section 3.3.1.

4.3.2 Training Details

We implement our model in Pytorch, and trained it on NVIDIA GeForce RTX 2080 Ti. In our experiments, we set the dimension of GC-LSTM hidden state to 200, and that of ELMo embedding to 1024. During training, ELMo parameters are not updated in order to reduce training time. The decoder hidden state dimension is set to 124 for Snips, and to 90 for ATIS. Dropout ratio is set to 0.5 to preventing overfitting, and the batch size is set to 32. The model is trained end-to-end using Adam optimizer (Kingma and Ba, 2014) to minimize the cross-entropy loss, with learning rate $= 1e^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 1e^{-9}$. Finally our graph convolution operation is approximated by $1st$-order Chebyshev polynomials.

4.3.3 Baselines

We adopt the three most popular evaluation metrics in SLU studies: slot filling using F1 score, intent prediction using accuracy, and sentence-level semantic frame parsing using whole frame accuracy. We compare our model with the following baselines:

Baselines models are from some typical works such as Joint Seq. (Hakkani-Tür et al., 2016), Attention BiRNN (Liu and Lane, 2016), Sloted-Gated (Goo et al., 2018), CAPSULE-NLU (Zhang et al., 2019), SF-ID Network (Niu et al., 2019), Key-value Memory (Wu et al., 2021), Unsupervised Transfer + ELMo (Siddhant et al., 2019) and Graph-LSTM (Zhang et al., 2020).

- Joint Seq. (Hakkani-Tür et al., 2016). This model proposed a RNN-LSTM architecture to jointly model domain detection, intent detection, and slot filling on multi-domains.
- Attention BiRNN (Liu and Lane, 2016). An attention-based neural network model for joint intent detection and slot filling was proposed to learn the relationship between slots and intents.
- Sloted-Gated (Goo et al., 2018). A gate mechanism was introduced to explore incorporating the intent information for slot filling in a joint model.
- CAPSULE-NLU (Zhang et al., 2019). This model proposed a capsule-based neural network model which accomplishes slot filling and intent detection via a dynamic routing-by-agreement schema. A rerouting schema was proposed to further synergize the slot filling performance using the inferred intent representation.
- SF-ID Network (Niu et al., 2019). This model introduced a SF-ID network to establish the interrelated mechanism for slot filling and intent detection tasks.
- Key-value Memory (Wu et al., 2021). This model explicitly modeled the long-term

slot context via a key-value memory network beneficial to both slot filling and intent detection.

- Unsupervised Transfer + ELMo (Siddhant et al., 2019). This model used Language Model (ELMo) Embeddings to take advantage of unlabeled data by learning contextualized word representations to efficiently transfer the knowledge from these unlabeled utterances to improve model performance on Spoken Language Understanding (SLU) tasks.

- Graph-LSTM (Zhang et al., 2020). This model attempted to address the limitation of sequential models by utilizing S-LSTM with a context-gated mechanism to learn the local context in dialogue utterances.

When doing the comparison, we adopt the reported results from those papers directly.

4.4 Experimental Results

4.4.1 Automatic Evaluation Results

Model	**ATIS Dataset**			**Snips Dataset**		
	Slot(F1)	**Intent(Acc)**	**Sent.(Acc)**	**Slot(F1)**	**Intent(Acc)**	**Sent.(Acc)**
Joint Seq.(Hakkani-Tür et al., 2016)	94.30	92.60	80.70	87.30	96.90	73.20
Attention BiRNN(Liu and Lane, 2016)	94.20	91.10	78.90	87.80	96.70	74.10
Sloted-Gated(Goo et al., 2018)	95.42	95.41	83.73	89.27	96.86	76.43
CAPSULE-NLU(?)	95.20	95.0	83.40	91.80	97.30	80.90
SF-ID Network(Niu et al., 2019)	95.58	96.58	86.00	90.46	97.0	78.37
Key-value Memory(Wu et al., 2021)	96.13	97.20	87.12	95.13	98.14	88.14
Unsupervised Transfer + ELMo(Siddhant et al., 2019)	95.62	97.42	87.35	93.90	**99.29**	85.43
Graph-LSTM(Zhang et al., 2020)	95.91	97.20	87.57	95.30	98.29	89.71
Graph-to-Seq	**96.37**	**97.88**	**88.69**	**95.89**	98.43	**90.57**

Table 4.1: SLU Performance evaluation results on ATIS and Snips datasets (%).

Table 4.1 shows the experimental results of our proposed model on ATIS and Snips datasets. On the single domain ATIS dataset, our model substantially outperforms all the baselines by a noticeable margin in all three aspects: slot filling (F1), intent detection (Acc) and sentence accuracy (Acc), demonstrating that explicitly modeling graph-structured dialogue context and the correlation between slots and intents can benefit SLU effectively via GC-LSTM. Compared with the prior joint work Graph-LSTM (Zhang et al., 2020), we achieve F1 score as 96.37% and intent Acc 97.88%, a significant improvement over 95.91% and 97.2%. This performance promotion signifies that our GC-LSTM can effectively model graph-structured dialogue context, and that our Graph-to-Seq framework captures long-term dependencies and models the correlation between slot filling and intent detection.

On the Snips dataset, our model also achieves good results in almost all cases especially on slot filling, which indicates our model has a better generalization capability than baseline models. Specifically, for slot filling, we achieve a F1 score of 95.89%, a salient enhancement compared with 95.3% (Zhang et al., 2020), and our sentence accuracy reaches at 90.57%. The gain further demonstrates the effectiveness of our proposed Graph-to-Seq framework. Although the intent Acc. of Unsupervised Transfer + ELMo model is slightly higher than ours, this is at the cost of slot filling performance.

Generally, the ATIS dataset is a simpler SLU task than Snips, so the room to be improved is relatively small. However, we still obtain noticeable improvement and set a new state-of-the-art result. On the other hand, Snips dataset is more complex crossing multiple domains. Thus, it is not surprising that most of baseline models are doing poorly especially on slot filling. Surprisingly, our model achieves a great performance jump especially on slot filling. Again we attribute this to our Graph-to-seq framework and GC-LSTM.

4.4.2 Ablation Study

In this section, we explore how each component contributes to our full model by conducting three important scenarios: (1) **With only GC-LSTM**. In this case, we directly compare the performance between GC-LSTM and S-LSTM (Zhang et al., 2020) to verify the effectiveness

Model	ATIS Dataset			Snips Dataset		
	Slot(F1)	**Intent(Acc)**	**Sent.(Acc)**	**Slot(F1)**	**Intent(Acc)**	**Sent.(Acc)**
Graph-LSTM baseline (Zhang et al., 2020)	95.91	97.20	87.57	95.30	98.29	89.71
With only GC-LSTM	96.21	97.01	87.68	95.40	98.43	89.71
With GC-LSTM and LSTM Decoder no decoding intent	96.30	96.75	87.23	95.68	98.0	89.57
Our full model: Graph-to-Seq	**96.37**	**97.88**	**88.69**	**95.89**	**98.43**	**90.57**

Table 4.2: Feature ablation study on our proposed model on ATIS and Snips datasets (%).

of our GC-LSTM. (2) **With GC-LSTM and LSTM decoder but without decoding intent**. This is to verify the effectiveness of a LSTM decoder. (3) **With full Graph-to-Seq framework**.

Table 4.2 shows the SLU performance variance on these scenarios. First, we only consider GC-LSTM to model spatio-temporal features of dialogue utterances by replacing S-LSTM in Zhang et al. (2020). From Table 4.2, we can see that GC-LSTM does improve performance in almost all the cases especially on slot filling and shows its superiority over S-LSTM. The result can be interpreted as that GC-LSTM demonstrates great capability to model spatial and temporal dependencies among the dialogue context globally, whereas S-LSTM is more focused on the local context. We then apply a stand-alone LSTM decoder to perform slot decoding. It is noticeable that slot filling is enhanced further, though intent detection deteriorates. It is explainable that using an autonomous autoregressive decoder to generate slot tags token by token not only reduces decoding errors by conditioning on all previously generated tokens, but also alleviates the encoder's burden. However, this model unintentionally puts too much weight on slot filling with the sacrificing of intent detection performance, thus leading to this unbalanced result. Finally, when we jointly perform decoding of slot filling and intent detection, the performance further improves. We attribute this to the fact that sharing slot decoding context not only improves intent detection accuracy, but is also beneficial to slot filling by minimizing intent detection objective function via joint training. To sum up, in a joint SLU model leaning too much on one task potentially worsens the other. Nevertheless, it is salient that our model achieves a trade-off to balance those two tasks.

Furthermore, we also study how the parameter time step in GC-LSTM impacts SLU per-

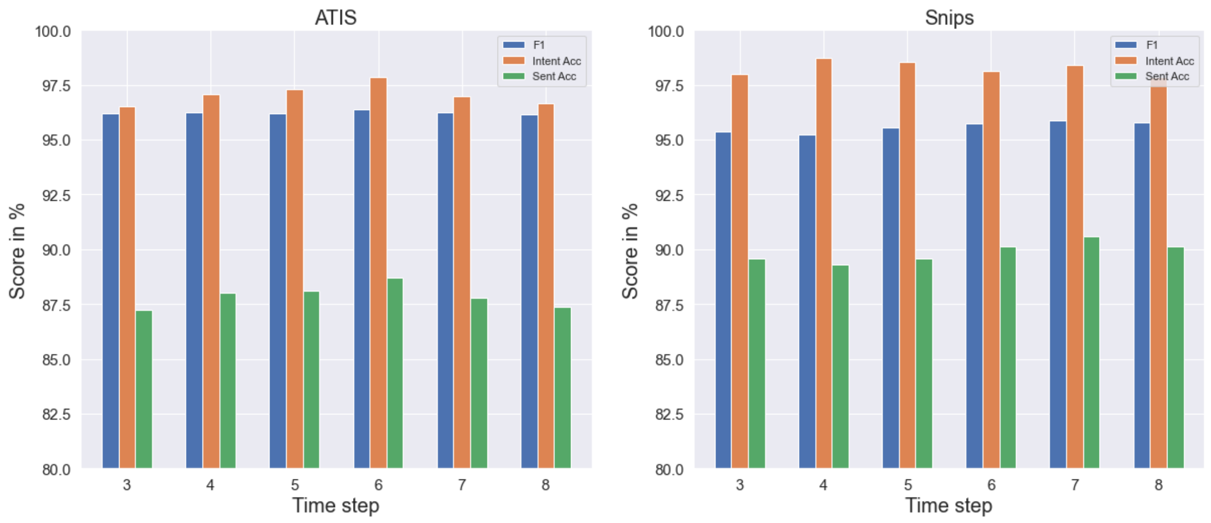

Figure 4.3: SLU performance on various time steps.

formance. Figure 4.3 shows the performance change with different time steps. It is easily observed that as the time steps go up, the sentence-level accuracy increases as well until reaching its peak. This is due to the message passing mechanism trying to enable word-level nodes to involve information spanning the whole dialogue graph. We find that the optimal time step for ATIS and snips datasets is 6 and 7, respectively.

4.4.3 Dialogue Dependency Graph vs N-gram Context Graph

We argue that modeling dialogue structural information by using our enhanced dependency graph is superior to the use of the n-gram context graphs. In order to verify this, we design some experiments to only replace our dialogue dependency graph with n-graph context graph with window size 1, 2 and 3. From Tables 4.3 and 4.4, it is noticeable that our dependency graph constantly outperforms the n-gram context graph with variable window sizes in all cases. We attribute this to the fact that modeling dialogue structural dependencies by our enhanced dependency graph captures spatial features globally, whereas the n-gram context graph is more focusing on limited local context. Anything outside the n-gram window has no impact on the decision being made.

Model	ATIS Dataset		
	Slot(F1)	Intent(Acc)	Sent.(Acc)
N-gram graph with window 1	96.12	97.09	87.46
N-gram graph with window 2	96.27	96.64	87.57
N-gram graph with window 3	96.28	96.42	87.35
Our dependency graph	**96.37**	**97.88**	**88.69**

Table 4.3: Performance comparison of dialogue dependency graph and n-gram context graph on ATIS (%).

Model	Snips Dataset		
	Slot(F1)	Intent(Acc)	Sent.(Acc)
N-gram graph with window 1	95.77	98.00	90.29
N-gram graph with window 2	95.61	97.71	89.71
N-gram graph with window 3	95.25	98.0	88.71
Our dependency graph	**95.89**	**98.43**	**90.57**

Table 4.4: Performance comparison of dialogue dependency graph and n-gram context graph on Snips (%).

4.4.4 Joint Model vs Separate Model

One of our main contributions is explicitly modeling the correlation and interaction of slots and intents by our GC-LSTM and joint decoding. Theoretically, this explicit interaction between them eventually promotes each other by achieving a trade-off. To verify this conclusion, we compare the SLU performance between the joint model and separate models. The former is our proposed model, whereas the latter is solely focusing on one task, thus without any interaction between intent detection and slot filling. It is easily observed from Table 4.5 that the joint model generally performs much better than two separate models. This further buttresses our claim.

Model	ATIS Dataset		Snips Dataset	
	Slot(F1)	Intent(Acc)	Slot(F1)	Intent(Acc)
Slot filling model	96.05	-	95.38	-
Intent detection model	-	97.20	-	98.0
Joint model	**96.37**	**97.88**	**95.89**	**98.43**

Table 4.5: Comparison between our joint model and separate models (%).

4.5 Conclusions

In this chapter, we propose a joint model to perform spoken language understanding with an augmented key-value memory to model slot context in order to capture long-term slot information. In addition, we adopt a gating mechanism to incorporate slot context information for intent classification to improve intent detection performance. Reciprocally, joint optimization promotes slot filling performance further by memory sharing between those two tasks. Experiments on two public datasets show the effectiveness of our proposed model.

Chapter 5

GraphMemDialogue: Learning End-to-End Dialogues

In this chapter, in addressing effectively integrating knowledge into dialogue generation systems, we propose a novel end-to-end learning framework to incorporate an external knowledge base (KB) and to capture the intrinsic semantics of the dialog history.

Effectively integrating knowledge into end-to-end task-oriented dialog systems remains a challenge. It typically requires to incorporate an external knowledge base (KB) and capture the intrinsic semantics of the dialog history. Recent research shows promising results by using Sequence-to-Sequence models, Memory Networks, and even Graph Convolutional Networks. However, current state-of-the-art models are less effective in integrating the dialog history and KB into task-oriented dialog systems in the following ways: **1**. The KB representation is not fully context-aware. The dynamic interaction between the dialog history and KB is seldom explored, which unfortunately are modeled separately. **2**. Both the sequential and structural information in the dialog history can contribute to capturing the dialog semantics, but they are not studied concurrently. In this chapter, we propose a novel Graph Memory

Network (GMN) based Seq2Seq model, GraphMemDialog, to effectively learn the inherent structural information hidden in dialog history, and to model the dynamic interaction between dialog history and KBs. We adopt a modified graph attention network to learn the rich structure representation in the dialog history, whereas the context-aware representation of KB entities are learnt by our novel GMN. To fully exploit this dynamic interaction, we design a learnable memory controller coupled with external KB entity memories to recurrently incorporate dialog history context into KB entities through a multi-hop reasoning mechanism. Experiments on three public datasets show that our GraphMemDialog model achieves state-of-the-art performance and outperforms strong baselines by a large margin, especially on datatests with more complicated KB information.

We first give an overview of learning end-to-end task-oriented dialogue systems. Then we present the architecture of our neural network based model in detail. Lastly, we discuss system learning, system evaluation and performance results.

5.1 Introduction

Task-oriented dialogue systems (TDSs), in contrast with chichat, aim at helping users complete a specific task with natural language, for example, inquiring about weather, reserving restaurants, and booking flights. In one specific domain, TDS takes dialog utterances and the knowledge base (KB) as input and produces responses by understanding dialog history, retrieving the most related KB entities, and generating readable sentences. Table 5.1 shows a multi-turn dialogue between a driver and an agent. The upper part is the KB information the dialogue needs to query, including point-of-interests (POIs), address, types, traffic information, and distances. In order to complete a navigation task, the dialogue system needs to query the KB and select the most relevant KB entities, such as *"valero"*, *"4 miles"*, and *"200 alester ave"*. Thus, effectively learning the representation of KB information and reasoning over it is very critical for TDSs. Traditionally, these dialog systems have been built as a pipeline, with modules including spoken language understanding (SLU), dialog state tracking, action selection and language generation Young et al. (2013b). However, pipelined dialog systems usually suffer from the credit assignment problem Yang et al. (2020) and easily lead to error propagation. Furthermore, they are not flexible enough to be adapted to new domains.

Distance	**Traffic info**	**Poi type**	**Address**	**Poi**
3 miles	heavy traffic	friend's house	580 van ness ave	tom's house
2 miles	moderate traffic	coffee or tea place	394 van ness ave	coupa
2 miles	no traffic	hospital	611 ames ave	palo alto medical foundation
5 miles	no traffic	hospital	214 el camino real	stanford express care
1 mile	heavy traffic	coffee or tea place	792 bedoin street	starbucks
2 miles	no traffic	chinese restaurant	842 arrowhead way	panda express
4 miles	heavy traffic	gas station	200 alester ave	valero

Role	**Turn**	**Utterance**
Driver	1	I need gas.
System	1	valero is 4 miles away.
Driver	2	what is the address?
System	2	valero is at 200 alester ave.

Table 5.1: A dialogue example with KB information on In-Car Assistant dataset in the navigation domain.

Recently, in order to address these issues, end-to-end TDSs Serban et al. (2016); Wen et al. (2017) with sequence-to-sequence model (Seq2Seq) have attracted attention due to great flexibility and good quality. The core idea of a Seq2Seq model is to leverage an encoder to directly map the dialog history and KB to a vector representation, which is then fed into a decoder to generate a response word by word. Later, memory networks (MemNN) are used to effectively incorporate KB information into the Seq2seq model Madotto et al. (2018); Zhong et al. (2018). Despite achieving promising results, these models are inherently weak at representing temporal dependencies between memories, thus ignoring the association between KB entities. Subsequently, Banerjee and Khapra (2019) first proposed to use Graph Convolutional Networks (GCNs) to capture the rich structural information hidden in the dialog history and the KB, and achieves big improvement compared with Seq2Seq models with attention. Yang et al. (2020) employed a new recurrent cell architecture to allow representation learning on graphs. However, these models still suffer from two major issues: **1) Learning of context-aware KB representation**. Although GCNs show promising results at capturing graph structure information inherent in the KB, current state-of-the-art models still fail to fuse meaningful dialog context semantics to the KB representation. Actually, our study shows that making it fully context-aware can significantly reduce the response errors especially on datasets with more complicated KB information. **2) Modeling sequential and structural dialog context concurrently**. Both type of information can be jointly rewarding to response generation by capturing distinct aspects of semantic information. Unfortunately, they have never been combined to promote performance in parallel for TDSs.

In this chapter, we propose a novel Graph Memory Network (GMN) based end-to-end task-oriented dialog model, GraphMemDialog. The proposed model learns fully context-aware KB representations integrated with both KB graph structure information and the dialog history semantics. Multi-hop reasoning further enables a deeper level of semantic modeling of KB entities. A multi-head graph attention network (GAT) is used to learn the intrinsic structural information in the dialog history, together with traditional RNNs, to jointly guide the response decoder to reduce response generation errors.

Our contributions are summarized as follows:

- We propose a novel GMN based end-to-end model to effectively incorporate external knowledge bases into TDSs by enabling fully context-aware KB entity representations. We design a learnable memory controller coupled with external KB entity memories to recurrently incorporate the dialog history context into KB entities through a multi-hop reasoning mechanism.

- We introduce a multi-head GAT to learn the intrinsic structural information in the dialog history to jointly guide the response decoder with the cooperation of traditional

RNNs.

- Experiments on three benchmark datasets (i.e., CamRest, In-Car Assistant, MultiWOZ 2.1) demonstrate that our GraphMemDialog outperforms the state-of-the-art models especially on In-Car Assistant and MultiWOZ 2.1 datasets with more complicated KB information.

5.2 Graph Memory Networks

Graph Memory Networks (GMNs) extend an end-to-end memory networks (Sukhbaatar et al., 2015) to have a structured dynamic memory organized as a graph of memory cells. Pham et al. (2018) proposed this new structure to perform molecular activity prediction. Lu et al. (2020) employed GMN framework to accomplish one-shot and zero-shot video object segmentation tasks. Khasahmadi et al. (2020) introduced a new memory layer for graph nerual networks which can learn hierarchical graph representations.

However, none of these GMNs work is to model the knowledge base in task-oriented dialog systems. Inspired by Pham et al. (2018) and Lu et al. (2020), we propose a novel GMN framework to effectively learn context-aware KB representations. To the best of our knowledge, our work is the first one that uses GMNs to incorporate the context-aware knowledge graph structure into an end-to-end model for task-oriented dialog systems.

5.3 Graph Memory Dialogue

Given a dialog between a user ($\mathbf{U}$) and a system ($\mathbf{S}$), the t-turned dialog utterance is represented as $(U_1, S_1), (U_2, S_2), ..., (U_t, S_t)$. Dialogs are associated with knowledge triples in the format of $(subject, relation, object)$ denoted as (h, r, t). The structural information of KB triples (h, r, t) is modeled as a knowledge graph $\mathcal{G}_k = (\mathcal{V}, \mathcal{E})$ with $h, t \in \mathcal{V}$ and $r \in \mathcal{E}$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of all entities and relations in $\mathcal{G}_k$, respectively. At the i^{th} dialog turn, our system input is dialog history $(U_1, S_1, ..., U_{i-1}, S_{i-1}, U_i)$ and the associated knowledge graph $\mathcal{G}_k$. The system output is to the generation of the next system response S_i word by word.

5.3.1 Model Overview

We propose a novel GMN framework to combine the merits of MemNNs and GCNs to effectively learn context-aware KB representations. Our proposed model is composed of three major components: a context graph encoder, a knowledge encoder based on GMN with multi-hopping reasoning, and a response decoder, as illustrated in Figure 5.1. The Context Graph Encoder learns a fixed-length vector to represent the dialog history both sequentially and structurally. We propose a graph encoder to encode the dialog history structural information, which is the dependency parsing graph of the sentences in the dialog history $\mathcal{G}_d$. Next the Knowledge Encoder encodes context-aware KB entity information by incorporating graph structure information and the dialog context semantics through our GMN and its multi-hop reasoning mechanism. Finally the Response Decoder generates the system response token-by-token, either by querying the knowledge graph, or by generating tokens from vocabularies under the constraint of the dialog and KB context. In the following sections we detail each component thoroughly.

5.3.2 Context Graph Encoder

Given that the dialog history has L utterances with each one containing T_i words, our context graph encoder encodes dialog context both sequentially and structurally via a hierarchical attention network (Yang et al., 2016) and a modified GAT Veličković et al. (2018). This

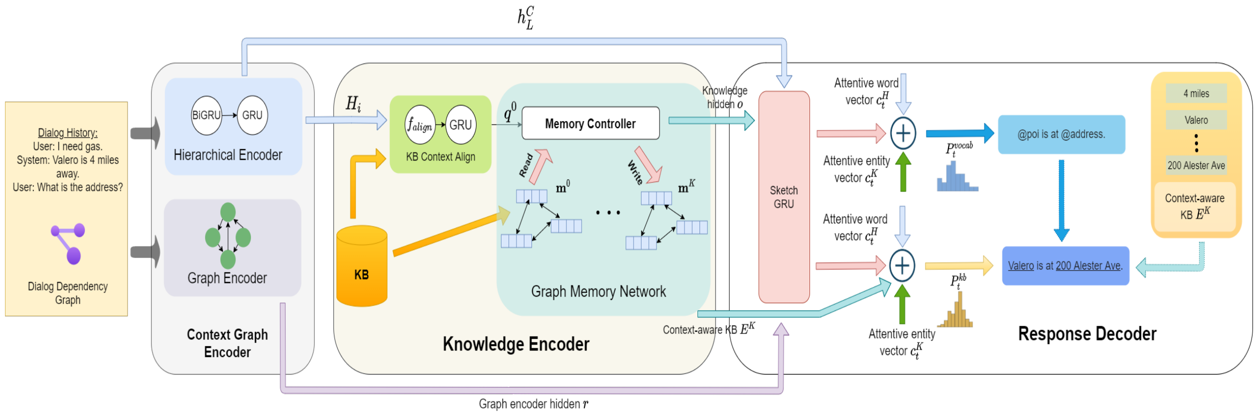

Figure 5.1: Graph Memory Dialogue Architecture.

approach is different than previous work in which dialog history is encoded either sequentially or structurally. We observe that the combination of those two inputs is beneficial to effectively capture the semantic context, especially for multi-turn dialog systems.

Hierarchical Attention Encoder We employ a hierarchical attention network to sequentially encode the dialog history. A bidirectional RNN is applied to learn representations of each word w_{it} with $t \in [1, T_i]$ in the i^{th} utterance by reading the input utterance forward and backward to produce context sensitive hidden states.We use BiGRU (Chung et al., 2014) to encode the dialog context into hidden states:

$$H_i = h_{i1}, ...h_{iT_i} = \textbf{BiGRU}(\phi^{emb}(w_{it}), h_{i(t-1)}) \tag{5.1}$$

where $\phi^{emb}(w_{it})$ is the embedding of the word w_{it}.

Then, we use self-attention mechanism to capture the contextual information for each token in order to get an interpretable utterance semantic representation as below:

$$u_{it} = \tanh(W_w h_{it} + b_w) \tag{5.2}$$

$$a_{it} = \frac{\exp(u_{it}^{\top} u_w)}{\sum_t \exp(u_{it}^{\top} u_w)} \tag{5.3}$$

$$\hat{s}_i = \sum_t a_{it} h_{it} \tag{5.4}$$

where W_w, b_w, u_w are trainable parameters of the model.

Finally, we use a GRU to encode the utterance vector $\hat{s}_i$:

$$h_i^C = \mathbf{GRU}(\hat{s}_i), i \in [1, L] \tag{5.5}$$

Graph Encoder In order to enable learning various relationships of words such as dependency relations, we first use the off-the-shelf parsing tool called Spacy[1] to extract dependency relation graph $\mathcal{G}_d$ among words in dialog history. To model word nodes and relations jointly, we employ a modified variant of graph attention network (GAT) (Veličković et al., 2018), which is enhanced to model multi-relational edges. In this way, modified GAT learns attention not only from the neighboring nodes, but also from edge features. This is important because dialog data contains rich edge information.

We first apply another BiGRU to process all the concatenated words in dialog history to get their contextual representation, which are then fed in our modified GAT:

$$h_t^G = \mathbf{BiGRU}(\phi^{emb}(w_t), h_{t-1}^G) \tag{5.6}$$

In the l-th layer of the modified GAT, the attention score between two neighboring words is obtained as follows:

$$\alpha_{ij} = \frac{\exp\left(\sigma\left(W[W_a h_i^{G^{(l)}} \| W_a h_j^{G^{(l)}} \| W_e {h_{i\to j}^{E}}^{(l)}]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\sigma\left(W[W_a h_i^{G^{(l)}} \| W_a h_k^{G^{(l)}} \| W_e {h_{i\to k}^{E}}^{(l)}]\right)\right)} \tag{5.7}$$

[1] https://spacy.io/

where ${h^E_{i\to j}}^{(l)}$ denotes the representation of the edge connecting word node i to its neighboring word node j. W_a and W_e are trainable word node and edge weights, whereas W is a single-layer feed-forward network parameter that computes the attention score. σ is the LeakyReLU activate function. Word hidden states are obtained by multi-head attention mechanism via averaging:

$$\widetilde{h}_t^G = \sigma\left(\frac{1}{N}\sum_{n=1}^{N}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^n W_n h_j^G\right) \tag{5.8}$$

where N is the number of heads, and W_n is the corresponding input linear transformation's weight matrix.

Finally we apply the same procedure on $\widetilde{h}_t^G$ as shown in eqs. (5.2) to (5.4) to obtain our final integrated graph context information:

$$o = \text{self-attention}\left(\widetilde{h}_t^G\right) \tag{5.9}$$

Contrary to BiGRU only capturing sequential associations between words, dependency relationship is supposed to learn the mutual interaction between head words and dependent words. Those two types of relationship should complement each other.

5.3.3 Knowledge Encoder

The knowledge encoder obtains a context-aware representation of each entity in the knowledge graph. We propose a novel graph memory network framework by extending a graph with multi-hop reasoning to model knowledge entities, through which entity dependencies can be well modeled and fused with dialog history context.

Context Alignment In order to make our knowledge entity context-aware, we first align the embedding of entity $e_j, j \in [1, E]$ with that of word w_{it} in the i^{th} utterance as shown in He et al. (2020), where E is the number of entities in the KB:

$$f^i_{align}(e_j) = \sum_t \alpha_{jit}\phi^{emb}(w_{it}) \tag{5.10}$$

$$\alpha_{jit} = \frac{\exp\left(u^\top_{jit}u_e\right)}{\sum_t \exp\left(u^\top_{jit}u_e\right)} \tag{5.11}$$

$$u_{jit} = \tanh\left(W_e[\phi^{emb}(e_j)\|\phi^{emb}(w_{it})] + b_e\right) \tag{5.12}$$

where W_e, b_e, and u_e are trainable parameters of the model and $\|$ denotes the concatenation. Then we pass the entity sequence to a GRU as follows for entity e_j:

$$f_{ji} = \text{GRU}\left([\phi^{emb}(e_j)\|f^i_{align}(e_j)]\right), i \in [1, L] \tag{5.13}$$

After the above processing, each entity has L representations corresponding to L utterances, so $F = \{f_{ji}\} \in \mathbb{R}^{L\times E\times d_e}$, where d_e is the entity embedding dimension.

Furthermore, to have a deeper integration of the dialog history, we perform another level of knowledge entity alignment with our sequential encoder BiGRU hidden state h_{it} as shown in Equation (5.1), because it captures the temporal dependency between words.

$$h^A_{ji} = \text{GRU}\left([f_{ji}\|f_{align}(f_{ji}, h_{it})]\right), j \in [1, E] \tag{5.14}$$

We take the entity representations under the L^{th} utterance from the output of the second alignment process as our initial context-aligned entity representations, namely:

$$E^A = \{h^A_{1L}, ...H^A_{EL}\} \in \mathbb{R}^{E\times d_e} \tag{5.15}$$

Graph Memory Network Our GMN is composed of an external graph memory and learnable controllers for memory reading and writing. The memory graph structure with multi-hop mechanism enables strong reasoning ability on knowledge entities. The controllers interact with memories using read and write operations to carry long-term context information and to encode new knowledge via slow updates of the weights. Through iterations, our GMN learns a general strategy to represent knowledge entities under a specific dialog context, making them quite context-aware.

We incorporate the semantics contained in the historical context into the KB entity memory slots. The memory is organized as a fully connected graph $\mathcal{G}_k = (\mathcal{V}, \mathcal{E})$, where node $m_i \in \mathcal{V}$ denotes i^{th} memory cell, which learns to represent entity e_i, and edge $\bar{e}_{ij} = (m_i, m_j) \in \mathcal{E}$ indicates the relation between entity e_i and e_j. The graph memory cells are initialized by the knowledge entity embedding $\{e_1, ..., e_E\}$. Subsequently these memory cells are augmented to capture the dialog history via controller writing.

Graph Memory Reading In order to effectively integrate context history into knowledge entities, we take context alignment output E^A as our initial state h_0^M to our GMN. A learnable read controller at each iteration step $k \in 1, ..., K$ interacts with graph memory by reading the content. m^k is a sum of all memory cells, weighted by the probability w_i^k. Formally Lu et al. (2020):

$$b_i^k = \frac{h_{k-1}^M \cdot m_i^{k-1}}{\|h_{k-1}^M\| \|m_i^{k-1}\|} \tag{5.16}$$

$$w_i^k = \frac{\exp(b_i^k)}{\sum_j \exp(b_j^k)} \tag{5.17}$$

$$m^k = \sum_i w_i^k m_i^{k-1} \tag{5.18}$$

Once reading memory content, the read controller updates its state as follows:

$$\begin{aligned} \widetilde{h}_k^M &= W_r^h h_{k-1}^M + U_r^h m^k \\ a_r^k &= \sigma(W_r^a h_{k-1}^M + U_r^a m^k) \\ h_k^M &= a_r^k \widetilde{h}_k^M + (1 - a_r^k) h_{k-1}^M \end{aligned} \tag{5.19}$$

where W_r^h, U_r^h, W_r^a, U_r^a are trainable parameters of the model. The udpate gate a_r^k controls how much previous hidden state h_{k-1}^M to be kept. In this way, the hidden state of the controller encodes both the KB entity memory and dialog history representations, hence context-aware.

Graph Memory Updating After we obtain a new query h_k^M, we need to update the graph memory with the new query input. At each step k, each memory cell is augmented by a learnable write controller, which is a function of previous memory state m_i^{k-1}, current query state h_k^M, and the states from all of its neighboring cells, namely:

$$m_i^k = f\left(m_i^{k-1}, h_k^M, \left(m_j^{k-1}\right)_{j \in \mathcal{N}(i)}\right) \tag{5.20}$$

where $\mathcal{N}_i$ is the neighbors of the node m_i. Following Pham et al. (2018), we calculate the summarized information c_i^k from neighboring entities as follows:

$$c_i^k = \sum_{j \in \mathcal{N}(i)} p_{i,j}^k \left[m_j^{k-1} \| \bar{e}_{i,j}^k\right] \tag{5.21}$$

where $\bar{e}_{i,j}^k$ is the relation feature vector between entities and $p_{i,j}^k$ is the weight of m_j, which indicates how important the node m_j towards m_i. $p_{i,j}^k$ can be learned, similarly to memory cell probabilities in the attentive reading as in Equations (5.16) and (5.17).

After aggregating the information from neighbors, the memory write controller updates the state of m_i as:

$$\begin{aligned} \widetilde{m}_i^k &= W_u^m h_k^M + U_u^m m_i^{k-1} + V_u^m c_i^k \\ a_u^k &= \sigma(W_u^a h_k^M + U_u^a m_i^{k-1} + V_u^a c_i^k) \\ m_i^k &= a_u^k \widetilde{m}_i^k + (1 - a_u^k) m^{k-1} \end{aligned} \tag{5.22}$$

The graph memory updating allows each memory cell to embed both the neighbor information and the dialog context information into its representation, making it fully context-aware. Moreover, by iteratively reasoning over the graph structure, each memory cell encodes the new query information and yields progressively improved representations. Those salient properties make our GMN overcome the shortness of traditional memory networks and also dwarf popular GCNs because of GMN's multi-hop reasoning.

Final GMN Outputs After K steps of iteration, we concatenate the output of context aligned KB entities and the final memory state m^K as our final knowledge entity representations:

$$E^K = \left[E^A \| m^K\right] \tag{5.23}$$

Also the first hidden state of h_K^M, that is $\mathbf{r} = h_K^M[0]$, is considered as our knowledge encoder hidden state to carry over KB context to the response decoder.

5.3.4 Response Decoder

The response decoder is conditioned on dialog sequential and structural context representation, and context-aware entity representation. Followed by Wu et al. (2019b), we use a sketch GRU to generate a sketch response that excludes slot values but includes sketch tags, which are all possible slot types starting with a special token, for instance, @distance. The sketch GRU learns to generate a dynamic dialogue action template. For example, instead of generating “Stanford shopping center is 2 miles away at 773 alger dr”, it produces “@poi is @distance away at @address”. At each decoding timestep, if a sketch tag is generated, we select an appropriate entity as the output word by querying entity representation. Otherwise, the output word is the word generated by the sketch GRU. For example, if “@poi” tag is generated, the words “Stanford shopping center” is selected from our KB entities to replace this tag as part of our final response.

The initial hidden state of the decoder h_0^D consists of context graph encoder hidden and knowledge encoder hidden output, which further constrains the decoding process under the current dialog context.

$$h_0^D = \sigma\left(W_d\left[h_L^C \| o \| r\right]\right) \tag{5.24}$$

where W_d is the trainable parameter. At each decoding time step t, the GRU takes the previously generated s_{t-1} and the previous hidden state h_{t-1}^D as the input and generates a new hidden state h_t^D as follows:

$$h_t^D = \text{GRU}\left(\phi^{emb}(s_{t-1}), h_{t-1}^D\right) \tag{5.25}$$

In order to handle long-term dependency, we again use an attention mechanism to dynamically determine the importance of each word in the dialog history and each entity in the KB. At each time step t, the decoder generates an attentive entity vector based on context-aware entity representation E^K as follows:

$$\alpha_{ti} = \frac{\exp\left({h_t^D}^\top \mathrm{W}_k e_i^K\right)}{\sum_j \exp\left({h_t^D}^\top \mathrm{W}_k e_j^K\right)} \tag{5.26}$$

$$c_t^K = \sum_{i=1}^{E} \alpha_{ti} e_i^K \tag{5.27}$$

Similarly, we generate an attentive word vector c_t^H based on H_L specified in Equation (5.1). Finally, the decoder generates two distributions, that is, a vocabulary distribution P_t^{vocab}, and a knowledge entity distribution P_t^{kb} to either select a vocabulary word or an entity word from the KB:

$$P_t^{vocab} = \text{Softmax}\left(W_v\left[h_t^D \| c_t^K \| c_t^H\right]\right) \tag{5.28}$$

$$P_t^{kb} = \text{Softmax}\left(E^{K^\top} W_{kb}\left[h_t^D \| c_t^K \| c_t^H\right]\right) \tag{5.29}$$

where W_v and W_{kb} are trainable parameters.

5.3.5 Joint Training

Following Wu et al. (2019b), we replace the slot values in the response S with sketch tags based on the provided entity table to create a sketch response $S^s = (s_1^s, ..., s_m^s)$. We use the standard negative log-likelihood loss to train the sketch GRU as:

$$\mathcal{L}_1 = \sum_{t=1}^{m} -\log(P_t^{vocab}(s_t^s)) \tag{5.30}$$

Similarly, we have another loss for our KB entities which eventually replace all the sketch tags to form a final response:

$$\mathcal{L}_2 = \sum_{t=1}^{m} -\log(P_t^{kb}(s_t^e)) \tag{5.31}$$

where $\{s_t^e\}$ represents the entity sequence. For the case when s_t is not an entity token, we train P_t^{kb} to produce a special token.

Finally the joint objective is formulated as the weighted-sum of these two loss functions using hyper-parameters α and β:

$$\mathcal{L}_\theta = \alpha\mathcal{L}_1 + \beta\mathcal{L}_2 \tag{5.32}$$

5.4 Experimental Setup

5.4.1 Datasets

To evaluate our proposed model, we conduct experiments on three widely used benchmark datasets: CamRest (Wen et al., 2016), In-Car Assistant (Eric and Manning, 2017b), and Multi-WOZ 2.1 (Qin et al., 2020).

CamRest This dataset contains dialogs in restaurant reservation domain, involving 676 multi-turn dialogs and having 5 turns on average per dialog Wen et al. (2016). It also has an average of 22.5 KB triples for every dialog. We divide this dataset into training/validaton/test sets with 406/135/135 dialogs, respectively, as Raghu et al. (2019) did.

In-Car Assistant It consists of 3,031 multi-turn dialogs in three distinct domains: weather (Wea.), navigation (Nav.), and schedule (Sch.) This dataset has an average of 2.6 turns. However the KB information is more compalicated than CamRest with an average of 62.3 triples for every dialog. Following Madotto et al. (2018), we divide the In-Car Assistant dataset into training/validaton/test sets with 2425/302/304 dialogs, respectively.

Multi-WOZ 2.1 By extending the Multi-WOZ (Budzianowski et al., 2018) to equip the corresponding KB to every dialog, this corpus is quite applicable for end-to-end response generation. This dateset contains three distinct domains: attraction (Att.), hotel (Hot.) and restaurant (Res.), with an average of 5.6 turns and 54.4 KB triples per dialog. Following how Qin et al. (2020) processed data, we deal with 1,839/117/141 dialogs for training/validation/test.

5.4.2 Training Details

We implement our model in Pytorch, which is trained on NVIDIA GeForce RTX 2080 Ti. In our experiments, we set all the embedding dimension and hidden units to 200 and batch size to 8. The model is trained end-to-end using Adam optimizer Kingma and Ba (2014) and learning rate annealing starts from $1e^{-3}$ to $5 \times 1e^{-5}$. Embeddings are randomly initialized and updated during training. For all the datesets, dropout ratio is set to 0.5, the number of our GAT's attention heads is set to 6, and the number of hops for GMN is set to be 2.

5.4.3 Baselines

We compare our proposed GraphMemDialog model with several representative works:

- **Seq2Seq+Attn** (Luong et al., 2015). This model adopted seq-to-seq with attention mechanism to improve neural machine translation.
- Mem2Seq (Madotto et al., 2018). Mem2Seq employed a memory network with multi-hop attention for attending over dialog history and KB triples.
- **GLMP** Wu et al. (2019b). A global memory encoder and a local memory decoder were proposed to share external knowledge. The encoder encoded dialogue history, modified global contextual representation, and generated a global memory pointer. The decoder generated responses by filtering the external knowledge via the global memory pointer.
- **DDMN** Wang et al. (2020). DDMN proposed two core components: dialog memory manager and KB memory manager. The dialog memory manager dynamically expanded the dialog memory turn by turn and kept track of dialog history with an updating mechanism, whereas the KB memory manager shared the structural KB triples throughout the whole conversation, and dynamically extracted KB information with a

memory pointer at each turn.

- **FG2Seq** He et al. (2020). FG2Seq encoded knowledge by considering inherent structural information of the knowledge graph and latent semantic information from dialog history.

- **MCL** Qin et al. (2021). This paper proposed a Meta Cooperative Learning framework for task-oriented dialog systems, consisting of an auxiliary KB reasoning task for learning meta KB knowledge, an auxiliary dialogue reasoning task for learning dialogue patterns, and a TDS task (primary task) that aims at not only retrieving accurate entities from KB but also generating natural responses.

When doing the comparison, we adopt reported results from those papers directly.

5.4.4 Automatic Evaluation Metrics

In order to have fair comparison with others' work, we adopt two most popular evaluation metrics in dialogue studies Zhong et al. (2018); Madotto et al. (2018); Qin et al. (2021).

1. **Bilingual Evaluation Understudy (BLEU)** Papineni et al. (2002). BLEU has been widely employed in evaluating sequence generation including machine translation, text summarization, and dialog systems. BLEU computes the n-gram overlap between the produced responses and gold ones.

2. **F1 Score (Entity F1)**. The entity F1 score is generally used to measure the system's capability of generating relevant entities to accomplish certain tasks by retrieving accurate entities from the provided KB. The entity F1 score is computed by micro-averaging the precision and recall over KB entities of the generated responses Qin et al. (2021).

Model	CamRest		In-Car Assistant					Multi-WOZ 2.1				
	BLEU	**Ent.F1**	**BLEU**	**Ent.F1**	**Sch.F1**	**Wea.F1**	**Nav.F1**	**BLEU**	**Ent.F1**	**Res.F1**	**Att.F1**	**Hot.F1**
Seq2Seq+Attn	7.7	21.4	9.3	19.9	23.4	25.6	10.8	4.5	11.6	11.9	10.8	11.1
Mem2Seq	13.5	33.6	12.6	33.4	49.3	32.8	20.0	6.6	21.6	22.4	22.0	21.0
GLMP	16.7	50.6	14.8	60.0	69.6	**62.6**	53.0	6.9	32.4	38.4	24.4	28.1
DDMN	19.3	58.9	17.7	55.6	65.0	58.7	47.2	12.4	31.4	30.6	32.9	30.6
MCL	20.1	59.2	17.2	60.9	70.6	**62.6**	**59.0**	13.6	32.6	34.4	30.2	29.8
Fg2Seq	20.2	62.1	16.8	61.1	73.3	57.4	56.1	13.5	36.0	40.4	41.7	30.9
GraphMemDialog	**22.3**	**64.4**	**18.8**	**64.5**	**75.9**	62.3	56.3	**14.9**	**40.2**	**42.8**	**48.8**	**36.4**

Table 5.2: Performance evaluation results on CamRest, In-Car Assistant, and Multi-WOZ 2.1 datasets.

5.5 Experimental Results

5.5.1 Automatic Evaluation Results

Table 5.2 shows the experiment results of the proposed model on CamRest, In-Car Assistant, and Multi-WOZ 2.1 datasets. From the table, we can see that our model substantially outperforms all the baselines by a noticeable margin on both BLEU score and entity F1, demonstrating that our context-aware graph memory network can benefit dialog response generation more effectively. On single domain CamRest dataset, compared with the best prior work Fg2Seq He et al. (2020), we achieve performance improvement by 10% on BLEU, and almost 4% on entity F1, respectively. The performance jump on BLEU score signifies that our decoder's generation error has been greatly reduced, whereas the gain on entity F1 indicates that our model can retrieve entities from the external knowledge data more accurately than those baselines. This demonstrates that our GraphMemDialog model can not only improve the dialog history context modeling by capturing the graph structure in dialogs via graph attention networks, but also effectively model the interaction between the dialog history and KB entities, making them fully context-aware.

On both In-Car assistant and Multi-WOZ 2.1 datasets, our GraphMemDialog also outperforms all the other baselines by a large margin both in BLEU score and entity F1, which indicates our model has a better generalization capability than baseline models. Our model outperforms Fg2Seq by 12% on BLEU score and 5.6% on entity F1 on In-Car Assistant, and by 10% on BLEU score and 12% on entity F1 on Multi-WOZ 2.1. The gain on entity F1 further demonstrates our GMN's great reasoning capability under different dialog history context, especially considering In-Car assistant and Multi-WOZ 2.1 have much more complicated KB information. Even though Fg2seq and MCL have already made a great advancement in performance, our GraphMemDialog still outperforms them by a large margin.

Table 5.3 reports some responses generated by GraphMemDialog and some baseline models. Compared with GLMP and FG2Seq, GraphMemDialog is more effective at carrying over dialog context to next turns and generating context-aware responses. For example, in the second turn, since the query is very short, GLMP and Fg2Seq tends to generate unrelated responses. GraphMemDialog shows strong capability to extract key entities, whereas GLMP fails to fill slot tag @weather_attribute it has produced. We attribute those merits mainly to our GMN's contributions.

Knowledge Base	
	today is monday downtown chicago monday hail downtown chicago monday low 50f downtown chicago monday high 70f

Role	**Turn**	**Utterance**
User	1	give me weather forecast for today.
System (Gold)	1	what city do you want the weather forecast for?
GLMP	1	what city do you want the weather for?
Fg2Seq	1	what city would you like to hear the forecast for?
GraphMemDialog	1	what city are you interested in?
User	2	downtown chicago, please.
System (Gold)	2	**today** in **downtown chicago** there should be **hail** with a high of **70f**.
GLMP	2	it will be **downtown chicago** in **today** monday.
Fg2Seq	2	what would you like to know about **today**?
GraphMemDialog	2	**today** in **downtown chicago** it will be **hail** today, and **hail** with a low of 50f and a high of **70f**.

Table 5.3: Responses generated by GraphMemDialog and some baseline models on In-Car Assistant dataset. The gold entities in each response are highlighted in bold.

5.5.2 Ablation Study

Model	CamRest		In-Car Assistant		Multi-WOZ 2.1	
	BLEU	**Ent.F1**	**BLEU**	**Ent.F1**	**BLEU**	**Ent.F1**
GraphMemDialog	**22.3**	**64.4**	**18.8**	**64.5**	**14.9**	**40.2**
w/o GMN	20.8	56.2	16.7	52.0	14.1	31.2
w/o GAT	21.2	62.4	18.8	63.6	14.2	39.0
w/o Both	20.2	54.8	16.4	50.6	12.9	30.8

Table 5.4: Ablation results of GraphMemDialog on CamRest, In-Car Assistant, and Multi-WOZ 2.1 datasets.

In this section, we explore how each component contributes to our full model. We conduct some ablation tests by removing GMN (w/o GMN), and modified GAT (w/o GAT). Table 5.4 shows the performance change. Firstly, if we only remove GMN, which means no KB structural information and no iterative interaction between dialog hisotry and KB involved, the performance degrades dramatically, especially on entity F1. This further demonstrates that our GMN makes a major contribution to our performance improvement. This is due to the fact that GMN not only learns graph structure inherent in KB, but also models the interaction between the dialog history and KB effectively as fully context-aware, enhancing the possibility to retrieve the most relevant entities from the KB. Next, if we only remove modified GAT, it is noticeable that the performance is degraded, but not significantly. Also we can observe that GAT is more helpful improving BLEU score than lifting entity F1. We attribute this to the fact that GAT is mainly for capturing structural information in the dialog history which helps reduce decoder generation errors. Finally, if we remove both modules, it is not surprising that the performance drops dramatically. This verifies that our GraphMemDialog model makes great contribution to model context-aware external knowledge base, and capturing the structural information in dialog history.

5.5.3 Comparison with Conventional GCNs

Our proposed GMN shows improvement on modeling context-aware KB. Due to its multi-hop reasoning, we expect it to be superior to conventional GCNs. In order to verify this, we design some experiments to replace our GMN with two widely-used GCNs: Relational Graph Convolutional Network (RGCN) Schlichtkrull et al. (2017), and Composition-based Multi-relational Graph Convolutional Network (COMPGCN) Vashishth et al. (2020). To be fair, we carefully choose GCNs which can encode relations as well, since our GMN does that. From Table 5.5, it is noticeable that our GMN outperforms COMPGCN and RGCN on both BLEU score and entity F1 on all three datasets in almost all cases. We attribute this to our GMN's multi-hop reasoning capability that effectively fuses dialog history context into KB entity representation, making it fully context-aware.

Model	CamRest		In-Car Assistant		Multi-WOZ 2.1	
	BLEU	**Ent.F1**	**BLEU**	**Ent.F1**	**BLEU**	**Ent.F1**
GraphMemDialog	**22.3**	**64.4**	**18.8**	**64.5**	**14.9**	40.2
COMPGCN	21.1	60.7	18.0	61.8	13.7	**40.3**
RGCN	21.3	60.3	18.1	63.4	14.1	36.3

Table 5.5: Performance comparison of GraphMemDialog with representative GCNs.

5.5.4 Error Analysis

To inspire future improvements, we inspect the generated responses manually. We empirically categorize our generation errors into three classes as shown in Table 5.6.

We find that a majority of generation errors fall into **type1** and **type2**. This may be attributed to the training of sketch GRU in the decoder to generate skewed response templates. On the other hand, **type3** is very limited, which further proves that our GMN shows great reasoning capability to select the most relevant KB entities mostly.

Error Type	Response
Error Type1: Omit entities	Gold System Response: you have two meeting scheduled , one at 4pm in conference room 102 to discuss the company picnic , and one at 6pm in 100 conference room to go over the quarterly report. Generated Response: you have two meeting scheduled one on monday at 4pm to discuss the company picnic.
Error Type2: Duplicate KB entities	Gold System Response: in downtown chicago it will be humid on monday , snow on tuesday , dry on wednesday , frost on thursday , drizzle on friday , clear skies on saturday and sunday. Generated Response: in downtown chicago , it will be humid on monday , snow on tuesday , dry on wednesday , dry on thursday , drizzle on friday , clear skies on saturday, clear skies on saturday clear skies on saturday , clear skies on saturday , clear skies on sunday.
Error Type3: Wrong slot filling	Gold System Response: it is not hailing in manhattan , the forecast says it it is dry today. Generated Response: it is not dry in manhattan right now.

Table 5.6: Categorized error types in our GraphMemDialogue.

5.6 Conclusion

In this chapter, we present a Graph Memory Network based end-to-end model for task-oriented dialog systems. GraphMemDialog models context-aware KB entities, and learns graph structure information hidden in dialog history and KBs. To fully fuse dialog context information into the KB, we design a learnable memory controller coupled with an external KB entity memory to recurrently incorporate the dialog history context into KB entities via a multi-hop reasoning mechanism. A modified GAT is employed to effectively capture graph structure information inherent in dialog history. Experiments on three public datasets show the effectiveness of our proposed model and achieve state-of-the-art results.

Chapter 6

Conclusion and Future Work

In this dissertation, we focus primarily on optimizing task-oriented dialogue systems with deep learning models. We firstly gave an overview of mainstream methodologies on solving some core research problems in task-oriented dialogue systems, such as spoken language understanding, end-to-end dialogue generation, etc. We have identified three major limitations of start-of-the-art neural models:

- RNN-based nerual models are inherently unstable over long time sequences because the memories are the RNN hidden states, and tend to focus more on short-term memories and forcefully compress historical records into one hidden state vector.
- Dominant RNNs focus primarily on modeling sequential dependencies, and thus rich graph structure information hidden in the dialogue context is ignored.
- Effectively incorporating external knowledge into end-to-end task-oriented dialogue systems still remains a challenge. Current state-of-the-art approaches fail to effectively model context-aware and graph-structured dialogue knowledge.

To effectively address these limitations, we explored new ways to model long-term dialogue context and to learn graph-structured representations of the dialogue history and the external knowledge bases, achieving state-of-the-art performance in spoken language understanding and end-to-end dialogue generation. In this chapter, we conclude our thesis and discuss future directions to continue task-oriented dialogue research.

6.1 Conclusions

In Chapter 3, we showed how we leveraged key-value memory networks to track long-term slot context in order to compensate the weakness of RNNs to model long word sequences. We first pointed out that current joint learning models ignored two important facts: **1.** Long-term slot context was not traced effectively, which is crucial for future slot filling. **2.** Slot tagging and intent detection could be mutually rewarding, but bi-directional interaction between slot filling and intent detection remained seldom explored. Then we proposed a novel approach to model long-term slot context and to fully utilize the semantic correlation between slots and intents. We adopted a key-value memory network to model slot context dynamically and to track more important slot tags decoded before, which are then fed into our decoder for slot tagging. Furthermore, gated memory information was utilized to perform intent detection, mutually improving both tasks through global optimization. Experiments on benchmark ATIS and Snips datasets showed that our model outperforms mainstream methods, especially for the slot filling task.

In Chapter 4, we described our graph-to-sequence learning framework to model the graph-structured features in the dialogue utterances, and to jointly decode intent detection and slot filling. Although RNN-based neural models showed promising results by jointly learning of these two tasks, dominant RNNs were primarily focusing on modeling sequential dependencies. Rich graph structure information hidden in the dialogue context was seldomly explored. We proposed a novel Graph-to-Sequence model to tackle the spoken language understanding problem by modeling both temporal dependencies and structural information in a conversation. We introduced a new Graph Convolutional LSTM (GC-LSTM) encoder to learn the semantics contained in the dialogue dependency graph by incorporating a powerful graph convolutional operator. Our proposed GC-LSTM not only captured the spatio-temporal semantic features in a dialogue, but also learned the co-occurrence relationship between intent detection and slot filling. Furthermore, a LSTM decoder was utilized to perform final decoding of both slot filling and intent detection, which mutually improves both tasks through global optimization.

In Chapter 5, we introduced an end-to-end generative model on dialogue response generations. Firstly, we identified that current state-of-the-art models were less effective in integrating the dialog history and KB into task-oriented dialog systems in the following ways: **1**. The KB representation was not fully context-aware. The dynamic interaction between the dialog history and KB was seldom explored, which unfortunately were modeled separately. **2**. Both the sequential and structural information in the dialog history could contribute to capturing the dialog semantics, but they were not studied concurrently. We proposed a novel Graph Memory Network (GMN) based Seq2Seq model, GraphMemDialogue, to effectively learn the inherent structural information hidden in dialog history, and to model the dynamic interaction between dialog history and KBs. We adopted a modified graph attention network to learn the rich structure representation in the dialog history, whereas the context-aware representation of KB entities were learnt by our novel GMN. To fully exploit this dynamic interaction, we designed a learnable memory controller coupled with external KB entity memories to recurrently incorporate dialog history context into KB entities through a multi-hop reasoning mechanism. Experiments on three public datasets showed that our GraphMemDialog model achieves state-of-the-art performance and outperforms strong baselines by a large margin, especially on datatests with more complicated KB information.

6.2 Future Work

With these improvements, we created a task-oriented learning framework to learn the long-term dialogue context and the graph-structured representations of the dialogue history and the external knowledge base. There are still unexplored areas and methodologies that are promising in task-oriented dialogue systems. These are different paths that may be taken to expand the work of this dissertation.

6.2.1 Joint pre-training of Language Models and Knowledge Bases

Data scarcity is a long-standing and crucial challenge that hinders the rapid iteration of task-oriented dialogue systems across multiple domains. Unfortunately, current approaches to build task-oriented dialogue systems still require a substantial amount of labelling and therefore are labor-intensive. On the other hand, large-scale pre-trained language models such as BERT (Devlin et al., 2019) and GPT (Budzianowski and Vulic, 2019) have achieved great success on a variety of NLP tasks, proving the effectiveness of pre-trained models. Gu et al. (2020) proposed a tailored pre-trained model for task-oriented dialog generation to solve data scarcity problem. However, seldom work has explored to jointly pre-train language understanding models and knowledge bases. As indicated in Chapter 5, effectively integrating KB into task-oriented dialogue systems is very critical to dialogue generations and understanding of KBs requires related dialogue contexts. Thus it is promising to co-train language models and knowledge bases to generate knowledge-aware language models especially for task-oriented dialogue systems. Again, our graph memory networks proposed in Chapter 5 can be utilized to model the interaction between dialogue history and knowledge bases. Together with BERT or GPT language model framework, a knowledge-aware pre-trained language model can be potentially built, thereby improving the task-oriented dialogue system performance. This remains one of our primary research directions in the near future.

6.2.2 Transfer Reinforce Learning of Graph-structured Dialogue Policies

In this thesis, our learning-based task-oriented dialogue systems focus primarily on one single domain, and no crossing domains is involved in a conversation. However, in real dialogues, people might jump from one topic to another and cover many different tasks in the same dialogue. Designing of task-oriented conversations capable of handling complex tasks trained to multiple domains is quite an onerous task. Lack of high quality, domain specific conversational data required to train dialogue policies is one of the biggest challenges for the success of any dialogue system. Transfer learning can be employed to boost faster and better learning of a task-oriented dialogue system, and also reduce the dependence on huge amount of dialogue data for a variety of domains. How to transfer the knowledge from one domain to anther is very significant and promising in term of designing a realistic task-oriented dialogue system. Furthermore, reinforce learning shows great success in learning strategies for managing multi-domain, multi-intent system in an unified manner. Effectively learning of graph-structured dialogue policies based on transfer reinforce learning framework is another major research direction in the future.

In short, modeling graph-structured dialogue systems is more generic than main-stream sequential modeling by RNNs, and has not been fully explored yet. We believe extending current learning frameworks to grasp graph structures in task-oriented dialogue systems is beneficial to, but not limited to a variety of research areas, such as pre-trained language models, dialogue policy learning, etc.

Bibliography

Banerjee, S. and Khapra, M. M. (2019). Graph convolutional network with sequential attention for goal-oriented dialogue systems. *Transactions of the Association for Computational Linguistics*, 7:485–500.

Budzianowski, P. and Vulic, I. (2019). Hello, it's gpt-2 - how can i help you? towards the use of pretrained language models for task-oriented dialogue systems. In *EMNLP*.

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018). Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278.*

Chen, H., Liu, X., Yin, D., and Tang, J. (2017). A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.

Chen, W., Chen, J., Qin, P., Yan, X., and Wang, W. Y. (2019). Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3696–3709, Florence, Italy. Association for Computational Linguistics.

Chen, Y.-N. V., Hakkani-Tür, D., Tur, G., Gao, J., and Deng, L. (2016). End-to-end memory networks with knowledge carryover for multi-turn spoken language understanding. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*. ISCA.

Cheng, J., Dong, L., and Lapata, M. (2016). Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733.*

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014a). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078.*

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014).*

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling.

Coucke, A., Saade, A., Ball, A., Bluche, T., Caulier, A., Leroy, D., Doumouro, C., Gisselbrecht, T., Caltagirone, F., Lavril, T., et al. (2018). Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Cuayáhuitl, H., Keizer, S., and Lemon, O. (2015). Strategic dialogue management via deep reinforcement learning. *arXiv preprint arXiv:1511.08099*.

Cui, Z., Henrickson, K., Ke, R., Pu, Z., and Wang, Y. (2019). Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T., editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Eric, M. and Manning, C. D. (2017a). Key-value retrieval networks for task-oriented dialogue.

Eric, M. and Manning, C. D. (2017b). Key-value retrieval networks for task-oriented dialogue. *arXiv preprint arXiv:1705.05414*.

Goddeau, D., Meng, H., Polifroni, J., Seneff, S., and Busayapongchai, S. (1996). A form-based dialogue manager for spoken language applications. In *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*, volume 2, pages 701–704 vol.2.

Goo, C.-W., Gao, G., Hsu, Y.-K., Huo, C.-L., Chen, T.-C., Hsu, K.-W., and Chen, Y.-N. (2018). Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Gu, J., Wu, Q., Wu, C., Shi, W., and Yu, Z. (2020). A tailored pre-training model for task-oriented dialog generation. *arXiv preprint arXiv:2004.13835*.

Haffner, P., Tur, G., and Wright, J. H. (2003). Optimizing svms for complex call classification. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, volume 1, pages I–I.

Hakkani-Tür, D., Tur, G., Celikyilmaz, A., Chen, Y.-N. V., Gao, J., Deng, L., and Wang, Y.-Y. (2016). Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *Proceedings of The 17th Annual Meeting of the International Speech Communication Association (INTERSPEECH 2016)*. ISCA.

He, Z., He, Y., Wu, Q., and Chen, J. (2020). Fg2seq: Effectively encoding knowledge for end-to-end task-oriented dialog. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8029–8033.

Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990). The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Henderson, M. (2015). Machine learning for dialog state tracking: A review.

Henderson, M., Thomson, B., and Young, S. (2013). Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Huang, H.-Y., Choi, E., and tau Yih, W. (2019). Flowqa: Grasping flow in history for conversational machine comprehension.

Huang, X., Qi, J., Sun, Y., and Zhang, R. (2020). Mala: Cross-domain dialogue generation with action learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7977–7984.

Hyötyniemi, H. (1996). Turing machines are recurrent neural networks. *Proceedings of step*, 96.

Jurafsky, D. and Martin, J. H. (2020). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice-Hall, Upper Saddle River, NJ.

Khasahmadi, A. H., Hassani, K., Moradi, P., Lee, L., and Morris, Q. (2020). Memory-based graph networks.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Lee, C., Pieraccini, R., Tzoukermann, E., Gauvain, J., Levin, E., Wilpon, J., and Gorelov, Z. (1992). A speech understanding system based on statistical representation of semantics. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, volume 1, pages 193–196, Los Alamitos, CA, USA. IEEE Computer Society.

Lei, W., Jin, X., Kan, M.-Y., Ren, Z., He, X., and Yin, D. (2018). Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447, Melbourne, Australia. Association for Computational Linguistics.

Li, C., Li, L., and Qi, J. (2018). A self-attentive model with gate mechanism for spoken language understanding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3824–3833, Brussels, Belgium. Association for Computational Linguistics.

Liu, B. and Lane, I. (2016). Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Liu, F. and Perez, J. (2017). Gated end-to-end memory networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1–10.

Lu, X., Wang, W., Danelljan, M., Zhou, T., Shen, J., and Van Gool, L. (2020). Video object segmentation with episodic graph memory networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 661–679. Springer.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation.

Madotto, A., Wu, C.-S., and Fung, P. (2018). Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., et al. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539.

Mrkšić, N., Ó Séaghdha, D., Wen, T.-H., Thomson, B., and Young, S. (2017). Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

Niu, P., Chen, Z., Song, M., et al. (2019). A novel bi-directional interrelated model for joint intent detection and slot filling. *arXiv preprint arXiv:1907.00390*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *Proc. of NAACL*.

Pham, T., Tran, T., and Venkatesh, S. (2018). Graph memory networks for molecular activity prediction. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 639–644. IEEE.

Qin, B., Yang, M., Bing, L., Jiang, Q., Li, C., and Xu, R. (2021). Exploring auxiliary reasoning tasks for task-oriented dialog systems with meta cooperative learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13701–13708.

Qin, L., Che, W., Li, Y., Wen, H., and Liu, T. (2019). A stack-propagation framework with token-level intent detection for spoken language understanding. *arXiv preprint arXiv:1909.02188*.

Qin, L., Xu, X., Che, W., Zhang, Y., and Liu, T. (2020). Dynamic fusion network for multi-domain end-to-end task-oriented dialog. *arXiv preprint arXiv:2004.11019*.

Raghu, D., Gupta, N., and Mausam (2019). Disentangling language and knowledge in task-oriented dialogs.

Raymond, C. and Riccardi, G. (2007). Generative and discriminative algorithms for spoken language understanding. In *Eighth Annual Conference of the International Speech Communication Association*.

Sarikaya, R., Hinton, G. E., and Ramabhadran, B. (2011). Deep belief nets for natural language call-routing. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5680–5683.

Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. (2017). Modeling relational data with graph convolutional networks.

Seo, Y., Defferrard, M., Vandergheynst, P., and Bresson, X. (2016). Structured sequence modeling with graph convolutional recurrent networks.

Serban, I., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models.

Si, C., Chen, W., Wang, W., Wang, L., and Tan, T. (2019). An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1227–1236.

Siddhant, A., Goyal, A., and Metallinou, A. (2019). Unsupervised transfer learning for spoken language understanding in intelligent agents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4959–4966.

Simonovsky, M. and Komodakis, N. (2017). Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks.

Tur, G., Hakkani-Tür, D., and Heck, L. (2010). What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop*, pages 19–24.

Tur, G. and Mori, R. D. (2011). *Spoken language understanding: Systems for extracting semantic information from speech*. John Wiley & Sons.

Vashishth, S., Sanyal, S., Nitin, V., and Talukdar, P. (2020). Composition-based multi-relational graph convolutional networks.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017a). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017b). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks.

Walker, M. A., Rambow, O., and Rogati, M. (2002). Training a sentence planner for spoken dialogue using boosting. *Comput. Speech Lang.*, 16:409–433.

Wang, J., Liu, J., Bi, W., Liu, X., He, K., Xu, R., and Yang, M. (2020). Dual dynamic memory network for end-to-end multi-turn task-oriented dialog systems. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4100–4110, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Wang, Y., Shen, Y., and Jin, H. (2018). A bi-model based rnn semantic frame parsing model for intent detection and slot filling. *arXiv preprint arXiv:1812.10235*.

Wen, T.-H., Gašić, M., Kim, D., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015a). Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 275–284, Prague, Czech Republic. Association for Computational Linguistics.

Wen, T.-H., Gašić, M., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015b). Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal. Association for Computational Linguistics.

Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2016). A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Wen, T.-H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2017). A network-based end-to-end trainable task-oriented dialogue system.

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*.

Wu, C.-S., Madotto, A., Hosseini-Asl, E., Xiong, C., Socher, R., and Fung, P. (2019a). Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy. Association for Computational Linguistics.

Wu, C.-S., Madotto, A., Winata, G. I., and Fung, P. (2018). End-to-end dynamic query memory network for entity-value independent task-oriented dialog. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6154–6158. IEEE.

Wu, C.-S., Socher, R., and Xiong, C. (2019b). Global-to-local memory pointer networks for task-oriented dialogue.

Wu, J., Harris, I., and Zhao, H. (2021). Spoken language understanding for task-oriented dialogue systems with augmented memory networks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 797–806, Online. Association for Computational Linguistics.

Yan, Z., Duan, N., Chen, P., Zhou, M., Zhou, J., and Li, Z. (2017). Building task-oriented dialogue systems for online shopping. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Yang, S., Zhang, R., and Erfani, S. (2020). Graphdialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Yao, K., Zweig, G., Hwang, M.-Y., Shi, Y., and Yu, D. (2013). Recurrent neural networks for language understanding. In *Interspeech*, pages 2524–2528.

Ye-Yi Wang, Li Deng, and Acero, A. (2005). Spoken language understanding. *IEEE Signal Processing Magazine*, 22(5):16–31.

Young, S. (2006). Using pomdps for dialog management. In *2006 IEEE Spoken Language Technology Workshop*, pages 8–13. IEEE.

Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013a). Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013b). Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Zhang, C., Li, Y., Du, N., Fan, W., and Yu, P. S. (2019). Joint slot filling and intent detection via capsule neural networks.

Zhang, L., Ma, D., Zhang, X., Yan, X., and Wang, H. (2020). Graph lstm with context-gated mechanism for spoken language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9539–9546.

Zhang, X. and Wang, H. (2016). A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*, volume 16, pages 2993–2999.

Zhang, Y., Liu, Q., and Song, L. (2018). Sentence-state LSTM for text representation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 317–327, Melbourne, Australia. Association for Computational Linguistics.

Zhao, T. and Eskenazi, M. (2016). 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, Los Angeles. Association for Computational Linguistics.

Zhao, T., Lu, A., Lee, K., and Eskenazi, M. (2017). Generative encoder-decoder models for task-oriented spoken dialog systems with chatting capability. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 27–36, Saarbrücken, Germany. Association for Computational Linguistics.

Zhong, V., Xiong, C., and Socher, R. (2018). Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1458–1467, Melbourne, Australia. Association for Computational Linguistics.

Zhou, H., Huang, M., and Zhu, X. (2016). Context-aware natural language generation for spoken dialogue systems. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2032–2041.