Learning to Memorize in Neural Task-Oriented Dialogue Systems

by

Chien-Sheng Wu

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Master of Philosophy
in the Department of Electronic and Computer Engineering

June 2019, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Chien-Sheng Wu

June 2019

Learning to Memorize in Neural Task-Oriented Dialogue Systems

by

Chien-Sheng Wu

This is to certify that I have examined the above MPhil thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Prof. Pascale Fung, Thesis Supervisor

Prof. Bertram Shi, Head of Department

Thesis Examination Committee

1. Prof. Bertram Shi Department of Electronic and Computer Engineering
2. Prof. Pascale Fung Department of Electronic and Computer Engineering
3. Prof. Qifeng Chen Department of Computer Science and Engineering

Department of Electronic and Computer Engineering

June 2019

Acknowledgments

First and foremost I would like to thank my supervisor, Professor Pascale Fung. As an exchange student and a graduate student, Pascale provided me extremely valuable mentorship both in academics and in life. Through her careful guidance in the past two years, I entered the natural language processing field, found my own research interests, published my first top conference paper, and connected to the research and industrial communities. Pascale inspired me by her unique vision and passionate attitude. She trusted me like nobody else, and encouraged me to dream big and focus on big problems. I am extremely lucky and proud to have been advised by Pascale.

I appreciate Professor Bertram Shi and Professor Qifeng Chen for their time in being my thesis examination committee. I also had the fortune to gain research and development experience in the industry during my internship at Salesforce. I would like to thank Dr. Richard Socher for sharing his comprehensive views and giving me on-point advice. Thanks to my great mentor Dr. Caiming Xiong for guiding me to conduct impactful research and giving me the freedom to explore. I would like to express my appreciation to the whole Salesforce research team for the insightful research discussions and brainstorming.

The last two years at HKUST have been an amazing learning and playing experience with my lab friends. Andrea Madotto, thanks for all the in-depth discussions and close collaboration on dialogue research. Peng Xu, thanks for insightful consultation on affective computing. Genta Winata, thanks for leading the code-switching research group. Nayeon Lee, thanks for collaborating on the fact-checking project. Thanks also to Zhaojiang Lin, Jamin Shin, Jiho Park, Farhad Siddique, and many many others for the great exchange of research experience and ideas. Without their time discussing and working with me, my graduate life would not have gone this well, nor would my thesis. I wish them all the best in their research and look forward to meeting them in the near future.

Finally, and most importantly, I would like to send love to my whole family, Chia-Chin Wu, Mei-Chueh Wang, and Chun-Ping Wu, for their continuous and incomparable love, support, and encouragement. I also want to express my deepest gratitude to Nai-Wen Hu for making me a better person and motivating me to chase my dream. It was they who always stood behind me giving me endless support with all of their hearts. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Learning to Memorize in Neural Task-Oriented Dialogue Systems

by

Chien-Sheng Wu

Department of Electronic and Computer Engineering
The Hong Kong University of Science and Technology

Abstract

Dialogue systems are designed to communicate with a human via natural language and help people in many aspects. Task-oriented dialogue systems, in particular, aim to accomplish users goal (e.g., restaurant reservation or ticket booking) in minimal conversational turns. The earliest systems were designed with a large amount of hand-crafted rules and templates by experts, which were costly and limited. Therefore, data-driven statistical dialogue systems, including the powerful neural-based systems, have considerable attention over the last few decades to reduce the cost and provide robustness.

One of the main challenges in building neural task-oriented dialogue systems is to model long dialogue context and external knowledge information. Some neural dialogue systems are modularized. Although they are known to be stable and easy to interpret, they usually require expensive human labels for each component and have unwanted module dependencies. On the other hand, end-to-end approaches learn the hidden dialogue representation automatically and directly retrieve/generate system responses. They require much less human involvement, especially in the dataset construction. However, most of the existing models suffer from incorporating too much information into end-to-end learning frameworks.

In this thesis, we focus on learning task-oriented dialogue systems with deep learning models, which is an important research direction in natural language processing. We leverage the neural copy mechanism and memory-augmented neural networks to address the existing challenge of modeling and optimizing information in conversation. We show the effectiveness of

our strategy by achieving state-of-the-art performance in multi-domain dialogue state tracking, retrieval-based dialogue systems, and generation-based dialogue systems.

We first improve the performance of a dialogue state tracking module, which is the core module in modularized dialogue systems. Unlike most of the existing dialogue state trackers, which are over-dependent on domain ontology and lacking knowledge sharing across domains, our proposed model, the transferable dialogue state generator (TRADE), leverages its copy mechanism to get rid of ontology, share knowledge between domains, and memorize the long dialogue context. We also evaluate our system on a more advanced setting, unseen domain dialogue state tracking. We empirically show that TRADE enables zero-shot dialogue state tracking and can adapt to new few-shot domains without forgetting the previous domains.

Second, we utilize two memory-augmented neural networks, the recurrent entity network and dynamic query memory network, to improve end-to-end retrieval-based dialogue learning. They are able to capture dialogue sequential dependencies and memorize long-term information. We also propose a recorded delexicalization copy strategy to simplify the problem by replacing real entity values with ordered entity types. Our models are shown to surpass other retrieval baselines, especially when the conversation has a large number of turns.

Lastly, we tackle end-to-end generation-based dialogue learning with two successive proposed models, the memory-to-sequence model (Mem2Seq) and global-to-local memory pointer network (GLMP). Mem2Seq is the first model to combine multi-hop memory attention with the idea of the copy mechanism, which allows an agent to effectively incorporate knowledge base information into a generated response. It can be trained faster and outperforms other baselines in three different task-oriented dialogue datasets, including human-human dialogues. Moreover, GLMP is an extension of Mem2Seq, which further introduces the concept of response sketching and double pointers copying. We empirically show that GLMP surpasses Mem2Seq in terms of both automatic evaluation and human evaluation, and achieves the state-of-the-art performance.

Chapter 1

Introduction

1.1 Motivation and Research Problems

Dialogue systems, known as conversational agents or chatbots, can communicate with human via natural language to assist, inform and entertain people. They have become increasingly important in both research and industrial communities. Such systems can be split into two categories: chit-chat conversational systems and task-oriented dialogue systems, shown in Table 1.1, with the former designed to keep users company and engage them with a wide range of topics, and the latter designed to accomplish specific tasks, such as restaurant reservation or ticket booking. Task-oriented dialogue systems are required to understand user requests, ask for clarification, provide related information, and take actions. Unlike chit-chat systems, these systems also usually involve tracking the intentions of users, retrieving information from external databases, and planning for multi-turn conversations. In this thesis, we focus on task-oriented dialogue systems.

Usually, task-oriented dialogue systems have been built modularly, with modules for spoken language understanding (SLU) [1, 2], dialogue state tracking (DST) [3–5], dialogue management (DM) [6–8], and natural language generation (NLG) [9]. The SLU module performs semantic decoding, and passes the information to the DST module to maintain the states of the dialogue. The DM module subsequently takes dialogue states and produces dialogue actions for the next utterance. The dialogue action is then passed to the NLG module to obtain the final system response.

These components traditionally are designed with a large amount of hand written rules and templates by experts. Motivated by the need for a data-driven framework to reduce the cost of laboriously hand-crafting dialogue managers and to provide robustness against the errors

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

	Chit-Chat Dialogue Systems	**Task-oriented Dialogue Systems**
System	Hello, how are you today?	Hello, what can I help with you today?
User	I got the job promotion.	Can you help me book a dinner at 6?
System	That is awesome! Congrats!	No problem. Where should it be?
User	Thank you! Gonna celebrate it at bar tonight.	Nearby Taipei city for 4.
System	Definitely! Don't be too drunk haha.	What cuisine do you prefer?
User	But it is boring to drink alone.	Let's try Japanese food... Sushi maybe.
System	Ask more friends to join!	What do you think about this option: Sushi Bar?
User	They don't want to come out tonight...	Sounds good! Please make a reservation.
System	Why? Today is Sunday why not?	Done. Anything else I can help?
User	Because they gonna watch Game of Thrones.	No thank you.
System	Right people are crazy about that!	You are welcome. See you next time.

created by speech recognition, statistical dialogue systems were first introduced to include an explicit Bayesian model of uncertainty and optimize the policy, e.g., the partially observable Markov decision processes (POMDPs) [8]. Next, neural networks and deep learning, a specific set of algorithms for function approximation, are now transforming the natural language processing (NLP) field, e.g, neural statistical dialogue systems have considerable attention in recent years. With labelled data, a model can learn to minimize loss function through iterative gradient update (back-propagation) of its parameters.

Even though these neural statistical modularized systems are known to be stable and easy to interpret by combining domain-specific knowledge and a slot-filling technique, they usually have the following drawbacks: 1) Complicated human annotated labels are required. For example, SLU and DST need labels for every domain and slot, DM requires dialogue experts to label dialogue actions and slot information, and NLG needs comprehensive language templates or human rule; 2) Dependencies between modules are complex, which may result in serious error propagation. In addition, the interdependent modules in modularized systems may result in performance mismatch, e.g., the update of the down-stream module may cause other upper-stream modules to be sub-optimal. 3) Generalization ability to new domains or new slots is limited. With too specific domain knowledge in each module, it is difficult to extend the modularized architecture to a new setting or transfer the learned knowledge to a new scenario. 4) Knowledge base (KB) interpretation requires additional human defined rule. There is no neural memory architectures that are designed to learn and represent the database information.

End-to-end neural approaches are an alternative to the traditional modularized solutions for task-oriented dialogue systems. These approaches train the model directly on text transcripts of dialogues, learn a distributed vector representation of the dialogue states automatically and

Table 1.2: A knowledge base example for task-oriented dialogue systems.

Point-of-Interest	Distance	Traffic Info	POI Type	Address
Maizuru	5 miles	moderate traffic	japanese restaurant	329 El Camino Real
Round Table	4 miles	no traffic	pizza restaurant	113 Anton Ct
World Gym	10 miles	heavy traffic	gym and sports	256 South St
Mandarin Roots	5 miles	no traffic	chinese restaurant	271 Springer Street
Palo Alto Cafe	4 miles	moderate traffic	coffee or tea place	436 Alger Dr
Dominos	6 miles	heavy traffic	pizza restaurant	776 Arastradero Rd
Sushi Bar	2 miles	no traffic	japanese restaurant	214 El Camino Real
Hotel Keen	2 miles	heavy traffic	rest stop	578 Arbol Dr
Valero	3 miles	no traffic	gas station	45 Parker St

retrieve or generate the system response in the end. Everything in an end-to-end model is learned together with the joint objective functions. In this way, the models make no assumption on the dialogue state structure and additional human labels, gaining the advantage of easily scaling up. Specifically, using recurrent neural networks (RNNs) is an attractive solution, where the latent memory of the RNN represents the dialogue states. However, existing end-to-end approaches in task-oriented dialogue systems still suffer from the following problems: 1) They struggle to effectively incorporate dialogue history and external KB information into the RNN hidden states since RNNs are known to be unstable over long sequences. Both of them are essential because dialogue history includes information about users goal and external KB has the information that need to be provided (as shown in Table 1.2). 2) Processing long sequences using RNNs is very time-consuming, especially when encoding the whole dialogue history and external KB using an attention mechanism. 3) Correct entities are hard to generate from the predefined vocabulary space, e.g., restaurant names or addresses. Additionally, these entities are relatively important compared to the chit-chat scenario because it is usually the expected information in the system response. For example, a driver expects to get the correct address of the gas station rather than a random place, such as a gym.

We propose to augment neural networks with external memory and a neural copy mechanism to address the challenges of modeling long dialogue context and external knowledge information in task-oriented dialogue learning. Memory-augmented neural networks (MANNs) [10–12] can be leveraged to maintain long-term memory, enhance reasoning ability, speed up the training process, and strengthen the neural copy mechanism, which are all desired features to achieve better memorization of information and better conversational agents. A MANN writes external memory into its memory modules and uses a memory controller to read and write memories repeatedly. This approach can memorize external information and rapidly encode

long sequences since it usually does not require auto-regressive (sequential) encoding. Moreover, a MANN usually includes a multi-hop attention mechanism, which has been empirically shown to be essential in achieving high performance on reasoning tasks, such as machine reading comprehension and question answering. A copy mechanism, meanwhile, allows a model to memorize words and directly copy them from input to output, which is crucial to successfully generate correct entities. It can not only reduce the generation difficulty but is also more like human behavior. Intuitively, when humans want to tell others the address of a restaurant, for example, they need to "copy" the information from the internet or their own memory to their response.

In this thesis, we focus on *neural task-oriented dialogue learning* that can effectively *incorporate long dialogue context* and *external knowledge* information. We fist demonstrate how to memorize long dialogue context in dialogue state tracking tasks, including single-domain, multi-domain, and unseen-domain settings. Then we show how to augment neural networks with memory and copy mechanism to memorize long dialogue context and external knowledge for both retrieval-based and generation-based dialogue systems.

1.2 Thesis Outline

The rest of the thesis is organized as:

- Chapter 2 introduces the background and related work on task-oriented dialogue systems, sequence text generation, copy mechanisms, and memory-augmented neural networks.
- Chapter 3 presents the transferable dialogue state generator to effectively generate dialogue states with a copy mechanism. We further extend the model to multi-domain dialogue state tracking and unseen domain dialogue state tracking.
- Chapter 4 presents two memory-augmented neural networks, a recurrent entity network and dynamic query memory network, with recorded delexicalization copying for end-to-end retrieval-based dialogue learning. These two models are able to memorize long-term sequential dependency in dialogue.
- Chapter 5 introduces two proposed state-of-the-art dialogue generation models: memory-to-sequence and global-to-local memory pointer network. These two models combine multi-hop

attention mechanisms with the idea of the copy mechanism, which allows us to effectively incorporate long context information.

- Chapter 6 summarizes this thesis and discusses possible future research directions.

Chapter 2

Background and Related Work

In task-oriented dialogue systems, there are several terms that will be used frequently: dialogue history, domains, intentions, slots, slot values, states, and external knowledge base (KB). Dialogue history does not mean the whole conversational history between chatbots and users, instead it is the whole dialogue context in the current conversation. Domains are the topics of the current conversation, for example, restaurant domain is about restaurant reservation and taxi domain is about booking a taxi to somewhere. Intentions are the goals of each user utterance, for example, the intention of saying "*Is it going to rain tomorrow?*" is to check weather, and the intention of saying "*Schedule a party at 7 on Friday*" is to add schedule. Slots are the predefined variables in the dialogue that can be filled with all kinds of slot values. For example, a location slot can have its slot values such as Hong Kong and Taipei. Dialogue states are the semantic decoding results such as slot-value pairs. Tracking the states can be viewed as the Markov decision process. Lastly, external KB is the stored information that could be provided to users, which is large and dynamic. For example, there could be many possible restaurants that meet the criteria, and the weather at each city might change everyday.

In this chapter, we first review the existing modularized and end-to-end task-oriented dialogue systems. We then introduce sequence generation using recurrent neural networks in natural language processing, and attention-based copy mechanisms. Lastly, we cover the general idea of memory-augmented neural networks and end-to-end memory networks in detail.

2.1 Modularized Task-Oriented Dialogue Systems

The block diagram of a modularized task-oriented dialogue systems is shown in Figure 2.1. The automatic speech recognition (ASR) module first transcribes users' speech into text format. The speech transcriptions are passed to the spoken language understanding module, which

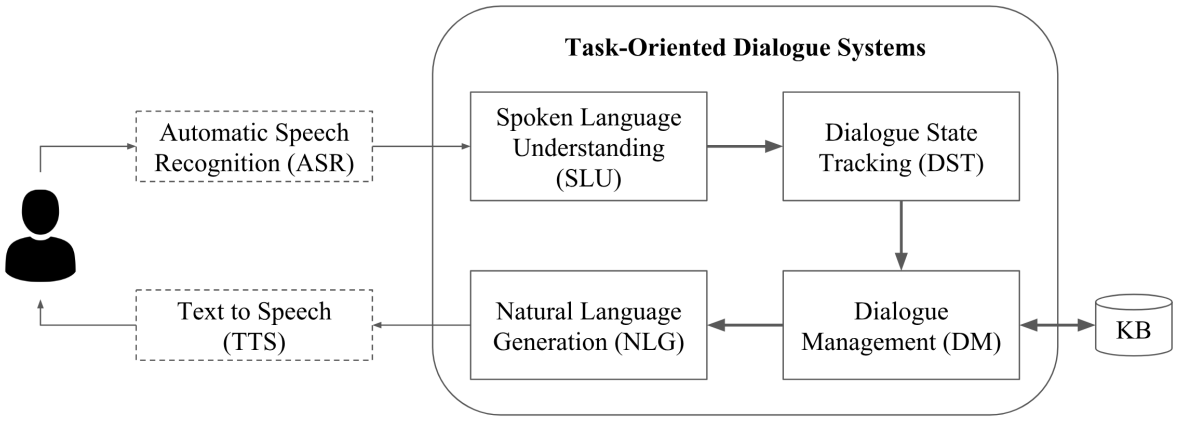


Figure 2.1: Block diagram of modularized task-oriented dialogue systems.

usually performs domain classification, intention detection, and slot-filling. The information extracted by the SLU module is next passed to the dialogue state tracking module, known as belief tracking, to maintain the states of the dialogue. The dialogue management module subsequently takes dialogue states and retrieved information from KB as input and produces dialogue actions for the next utterance. The dialogue action is then passed to the natural language generation module to obtain the natural language system response. Lastly, the text-to-speech (TTS) module transforms the text into speech and replies to users. In this thesis, we focus on all the modules except the ASR and TTS; that is, we focus on natural language processing in task-oriented dialogue systems.

2.1.1 Spoken Language Understanding

The SLU module [1, 2, 13–15] typically involves identifying users intentions and extracting semantic components from the user utterances. Intention detection [16, 17] can be framed as a semantic utterance classification problem. Given a sequence of words, the goal is to predict an intent class from intent candidates. Slot-filling [18–20] extracts semantic components by searching inputs to fill in values for predefined slots, which is a sequence labeling task that assigns a semantic label to each word. Slots are basically variables in utterances, which can have their own values but are by default empty. A slot and its value together can be viewed as a slot-value pair to represent the dialogue semantics.

2.1.2 Dialogue State Tracking

The DST module is a crucial component in task-oriented dialogue systems, where the goal is to extract user goals/intentions during a conversation as compact dialogue states. Given the extracted results from the SLU module, the DST module is usually expressed by a list of goal slots and the probability distribution of the corresponding candidate values. Early systems for dialogue state tracking [3–5] relied on hand-crafted features and a complex domain-specific lexicon and domain ontology. To reduce human effort and overcome the limitations of modeling uncertainties, learning-based dialogue state trackers have been proposed in the literature [21–26].

2.1.3 Dialogue Management

The DM module [6–8] generates dialogue actions based on the dialogue states from the DST module and the retrieved information from the external KBs. The dialogue actions usually consist of a speech action (e.g., inform, request) and grounding information represented by slots and values. Moreover, several approaches use reinforcement learning methods to learn the policy for predicting the actions given by states [27–29].

2.1.4 Natural Language Generation

The most widely used method in NLG is template-based [9], which the response sentences are manually designed to be the output for all the possible dialogue actions in DM. The main advantages include the ease with which developers can control the system. However, building such a system is expensive when there are many dialogue actions, and it cannot handle the ones that not previously designed. To address these problems, machine learning-based NLG [30, 31] has been actively researched, where a corpus consisting of only dialogue actions and utterances.

2.2 End-to-End Task-Oriented Dialogue Systems

2.2.1 Overview

Unlike modularized systems that use a pipeline to connect each module, end-to-end neural systems are designed to train directly on text transcripts and the KB information. Inspired by the success of end-to-end training of chit-chat dialogue systems [32–35], researchers have recently

started exploring end-to-end solutions for task-oriented dialogue systems. RNNs, such as long short-term memory networks [36] (LSTMs) and gated recurrent units [37] (GRUs), play important roles in this direction due to their ability to create latent representations, avoiding the need for artificial labels. [38] proposed an end-to-end trainable neural dialogue model in which each system component is connected, and each module is trained separately. However, each component is pre-trained separately with different objective functions. [39] and [40] used end-to-end memory networks (MNs) [12] to learn utterance representations, and applied them to the next-utterance prediction task. Although the performance is similar to existing retrieval approaches, these models do not model the utterance dependencies in the memory. [41], meanwhile, considered utterances as a sequence of state-changing triggers and reduced the memory query to a more informed query through time. [42] proposed hybrid code networks (HCNs) that combine RNNs with domain-specific human rule, and [43] incorporated explicit dialogue state tracking into a delexicalized sequence generation.

Most related to this thesis, [44] is the first work that applied a neural copy mechanism into the sequence-to-sequence architecture for the task-oriented dialogue task. It showed that the performance of its generation model with the copy mechanism was competitive with other retrieval models on a simulated dialogue dataset. However, the authors encoded all the information, including the dialogue history and external KB, using a single RNN, which makes the encoding process time-consuming and makes it hard to learn good representations. This drawback is the motivations for our proposed architectures, which will be introduced in the following chapters on end-to-end dialogue response generation.

2.2.2 Recurrent Neural Networks

A recurrent neural network (RNN), shown in Figure 2.2, is a class of artificial neural network where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit dynamic temporal behavior. Differently from feed-forward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. A widely used RNN architecture is the Elman type RNN [45]. The 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. Then a non-linear and differentiable activation function is applied to the weighted sum of the input

vector and the previous hidden state. The hidden state at time t can thus be expressed as:

$$h_t = \sigma(Ux_t + Vh_{t-1} + b), \tag{2.1}$$

where σ is the non-linear activation function. In RNN training, parameters in the network can be optimized based on loss functions derived using maximum likelihood. Network parameters are updated using back-propagation through time (BPTT) 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 or mini-batch.

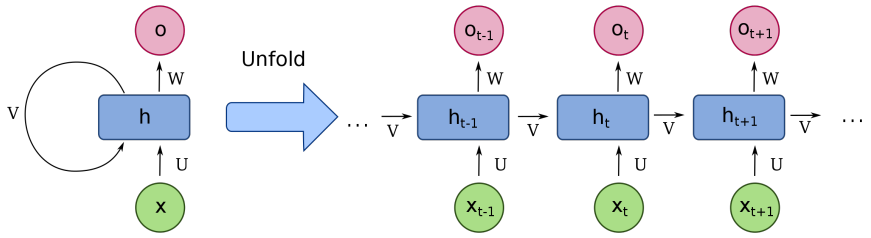


Figure 2.2: Unfolded basic recurrent neural networks.

Long Short-Term Memory and Gated Recurrent Units

Long short-term memory (LSTM) [36] and gated recurrent units (GRUs) [46] are variances of original recurrent neural network. The LSTM was followed by the Gated Recurrent Unit (GRU), and both have the same goal of tracking long-term dependencies effectively while mitigating the vanishing/exploding gradient problems. The LSTM does so via input, forget, and output gates: the input gate regulates how much of the new cell state to keep, the forget gate regulates how much of the existing memory to forget, and the output gate regulates how much of the cell state should be exposed to the next layers of the network. On the other hand, the GRU operates using a reset gate and an update gate. The reset gate sits between the previous activation and the next candidate activation to forget previous state, and the update gate decides how much of the candidate activation to use in updating the cell state. In this thesis, we utilize GRUs in most of the experiments because it has fewer parameters than LSTM and could be trained faster.

2.2.3 Sequence-to-Sequence Models

As shown in Figure 2.3, the most common and powerful conditional generative model for natural language is the sequence-to-sequence (Seq2Seq) model [46, 47], which is a type of encoder-decoder models. Seq2Seq models the target word sequence Y conditioned on given word sequence X, that is, $P(Y|X)$. The basic Seq2Seq uses an encoder RNN^{enc} to encode input X, and a decoder RNN^{dec} to predict words in output Y. Note that the encoder and decoder are not necessarily RNNs. Different kinds of sequence modeling approaches are possible alternatives. Let w_i^x and w_j^y be the i^{th} and j^{th} words in the source and target sequences, respectively. In most machine learning-based natural language applications, instead of representing words using one-hot vectors, words in vocabulary are usually represented by word embeddings, fixed-length vectors with real numbers. Embeddings can either be randomly initialized and learned through loss optimization or be pre-trained using a distributional semantics hypothesis [48, 49]. Afterwards, the source sequence is encoded by recursively applying:

$$h_0^{enc} = 0, \tag{2.2}$$

$$h_i^{enc} = \text{RNN}^{enc}(w_i^x, h_{i-1}^{enc}). \tag{2.3}$$

Then the last hidden state of the encoder $h_{|X|}^{enc}$ is viewed as the representation of X to initialize the decoder hidden state. The decoder then predicts the words in target Y sequentially via:

$$h_j^{dec} = \text{RNN}^{dec}(w_j^y, h_{j-1}^{dec}), \tag{2.4}$$

$$o_j = \text{Softmax}(W h_j^{dec} + b), \tag{2.5}$$

where o_j is the output probability for every word in the vocabulary at time step j. In addition, in order to make the model predict the first word and to terminate the prediction, special SOS (start-of-sentence) and EOS (end-of-sentence) tokens are usually padded at the beginning and the end of the target sequence.

2.2.4 Attention Mechanism

Although the standard Seq2Seq model is able to learn the long-term dependency in theory, it often struggles to deal with long-term information in practice. An attention mechanism [50, 51] is an important extension of the Seq2Seq models, mimicking the word alignment in statistical machine translation. The intuitive idea is that instead of solely depending on a fixed-length encoded vector, the attention mechanism allows the decoder to create dynamic encoded representations

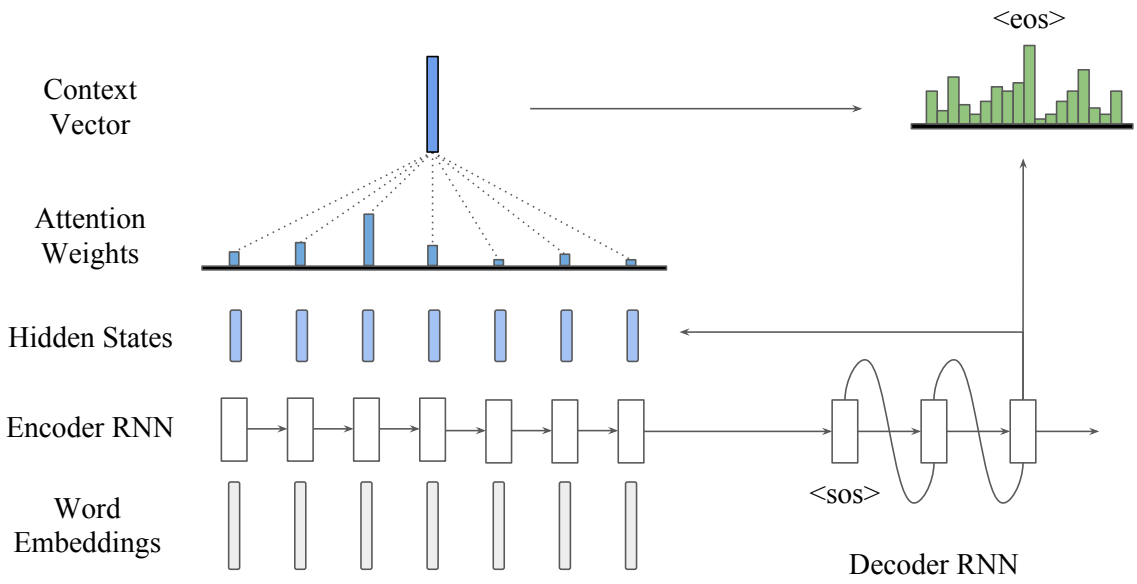


Figure 2.3: A general view of encoder-decoder structure with attention mechanism.

for each decoding time step by the weighted-sum encoded vector. Let $H^{enc} = \{h_1^{enc}, \ldots, h_{|X|}^{enc}\}$ be the hidden states of the encoder RNN^{enc}. Then at each decoding time step the decoder predicts the output distribution by

$$h_j^{dec} = \text{RNN}^{dec}(w_j^y, h_{j-1}^{dec}), \tag{2.6}$$

$$u_j^i = \text{Match}(h_i^{enc}, h_j^{dec}), \tag{2.7}$$

$$\alpha_j = \text{Softmax}(u_j), \tag{2.8}$$

$$c_j = \Sigma_i^{|X|}(\alpha_j^i h_i^{enc}), \tag{2.9}$$

$$o_j = \text{Softmax}(W[h_j^{dec}; c_j] + b). \tag{2.10}$$

The α vector is the attention score (probability distribution) computed by a matching function, which can be simple cosine similarity, linear mapping, or a neural network, as

$$\text{Match}(h_i, h_j) = \begin{cases} h_i h_j^\top, & \text{(dot)} \\ h_i W h_j^\top, & \text{(general)} \\ \tanh(W[h_i; h_j]). & \text{(concat)} \end{cases} \tag{2.11}$$

The c_j in Eq. (2.9) is the context vector in decoding time step j, which is the weighted-sum of the encoder hidden states based on the attention weights.

2.2.5 Copy Mechanism

A copy mechanism is a recently proposed extension of attention mechanisms. Intuitively, it encourages the decoder to learn how to "copy" words from the input sequence X. The main advantage of the copy mechanism is its ability to handle rare words or out-of-vocabulary (OOV) words. There are three common strategies to perform the copy mechanism: index-based, hard-gate, and soft-gate. Index-based copying usually produces the start and end positional indexes, and copies the corresponding text from the input source; hard-gate and soft-gate copying usually have two distributions, one over the vocabulary space and the other over the source text. Hard-gate copying uses a learned gating function to switch between two distributions, while soft-gate copying, on the other hand, combines two distributions into one with a learned scalar.

Pointer networks [52] were the first model to perform an index-based copy mechanism, which directly generates indexes corresponding to positions in the input sequence via:

$$o_j = \text{Softmax}(u_j). \tag{2.12}$$

In this way, the output distribution is the attention distribution, and the output word is the input word that has highest probability.

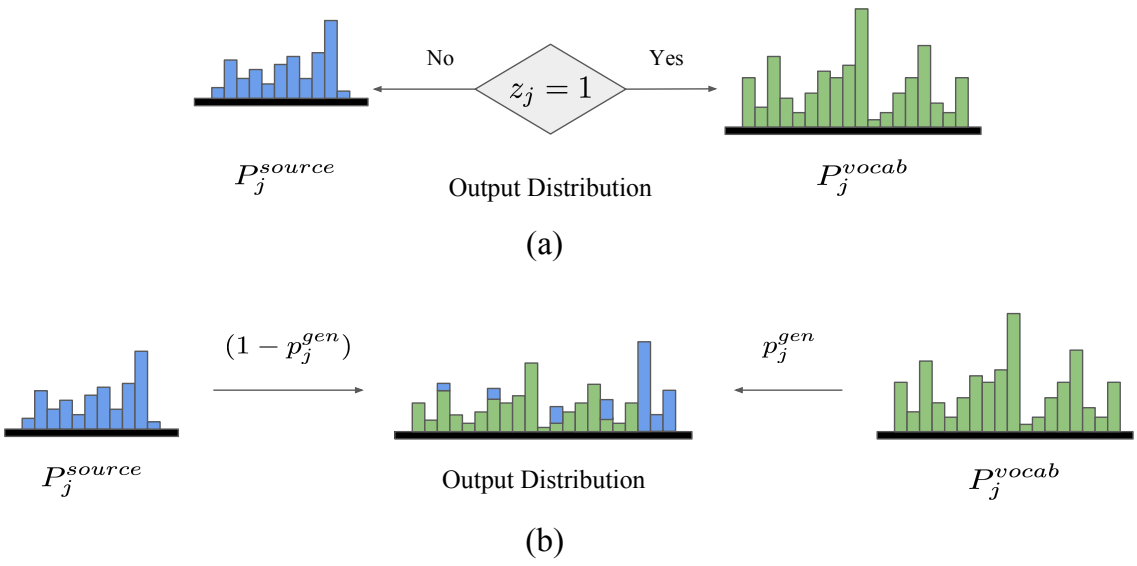


Figure 2.4: Copy mechanisms: (a) hard-gate and (b) soft-gate. The controllers z_j and p_j^{gen} are context-dependent parameters.

During decoding, hard-gate copying [53, 54], shown in Figure 2.4(a) uses a switch to select distributions, and picks up the output word from the selected distribution. The switching

probability z_j is modeled as a multi-layer perceptron with a binary output. The concept is similar to pointer networks but the decoder retains the ability to generate output words from the predefined vocabulary distribution:

$$P_j^{vocab} = \text{Softmax}(W[h_j^{dec}; c_j] + b), \tag{2.13}$$

$$P_j^{source} = \text{Softmax}(u_j), \tag{2.14}$$

$$z_j = \begin{cases} 1 & \text{if Sigmoid}(f(w_j^{dec}, h_{j-1}^{dec})) > 0.5, \\ 0 & \text{otherwise,} \end{cases}, \tag{2.15}$$

$$o_j = \begin{cases} P_j^{vocab} & \text{if } z_j = 1, \\ P_j^{source} & \text{otherwise.} \end{cases} \tag{2.16}$$

As shown in Eq. (2.16), the output distribution is dependent on z_j to switch between P_j^{vocab} and P_j^{source}. In this way, the model is able to generate an unknown word in the vocabulary by directly copying a word from the input to output. Usually, there are multiple objective functions combined to learn the output generation, at least one for the vocabulary space supervision and one for the gating function supervision.

The soft-gate copy mechanism [55–58], shown in Figure 2.4(b), on the other hand, combines the two distributions into one output distribution and generates words. Usually a context-dependent scalar p_j^{gen} is learned to weighted-sum the distributions:

$$p_j^{gen} = \text{Sigmoid}(W[h_t^{dec}; w_t^{dec}; c_j]), \tag{2.17}$$

$$o_j = p_j^{gen} \times P_j^{vocab} + (1 - p_j^{gen}) \times P_j^{source}. \tag{2.18}$$

In this way, the output distribution is weighted by the source distribution, given a higher probability that the word appears in the input. Note that words generated by the soft-gate copy mechanism are not constrained by the predefined vocabulary, i.e., the input unknown words will be concatenated with the vocabulary space. In this thesis, we adopt the hard-gate and soft-gate copying strategies.

2.3 Memory-Augmented Neural Networks

2.3.1 Overview

Although the recurrent approaches using LSTM or GRU have been successful in most cases, they may suffer from two main problems: 1) They struggle to effectively incorporate external

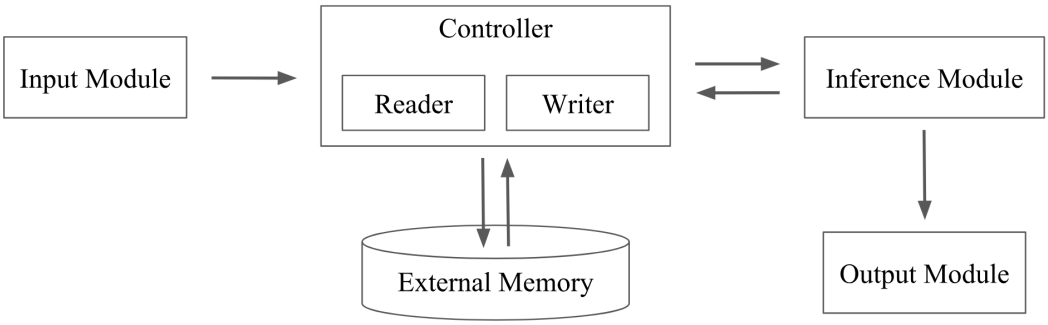


Figure 2.5: Block diagram of general memory-augmented neural networks.

KB information into the RNN hidden states [12], and they are known to be unstable over long sequences. 2) Processing long sequences one-by-one is very time-consuming, especially when using the attention mechanism. Therefore, in this section, we will introduce the general concept of memory-augmented neural networks (MANNs) [10–12, 59–63].

In Figure 2.5, we show the general block diagram of MANNs. The key difference compared to standard neural networks is that a MANN usually has an external memory that a controller can interact with. The memory can be bounded or unbounded, flat or hierarchical, read-only or read-write capable, and contains implicit or explicit information. The overall computation can be summarized as follows: The input module first receives the input and sends the encoded input to the controller. A controller reads relevant information from the memory, or does computation to store some information in the memory by writing. The controller sends the result of the computation to the inference module. Finally, the inference module does high-level computations and sends the results to the output module for general output.

2.3.2 End-to-End Memory Networks

Most related to this thesis, we now introduce end-to-end memory networks (MNs) [12] in detail. In Figure 2.6, [1] the left-hand side (a) shows how the model reads from and writes to the memory, and how the process can be repeated multiple times ("hops"), as shown in the right-hand side (b).

The memories of memory networks are represented by a set of trainable embedding matrices $E = \{ A^1, C^1, \ldots, A^K, C^K \}$, where each A^k or C^k maps tokens to vectors and K is the number of hops. There are two common ways of weight tying within the model, adjacent and layer-wise.

[1] The figure is from the original paper [12].

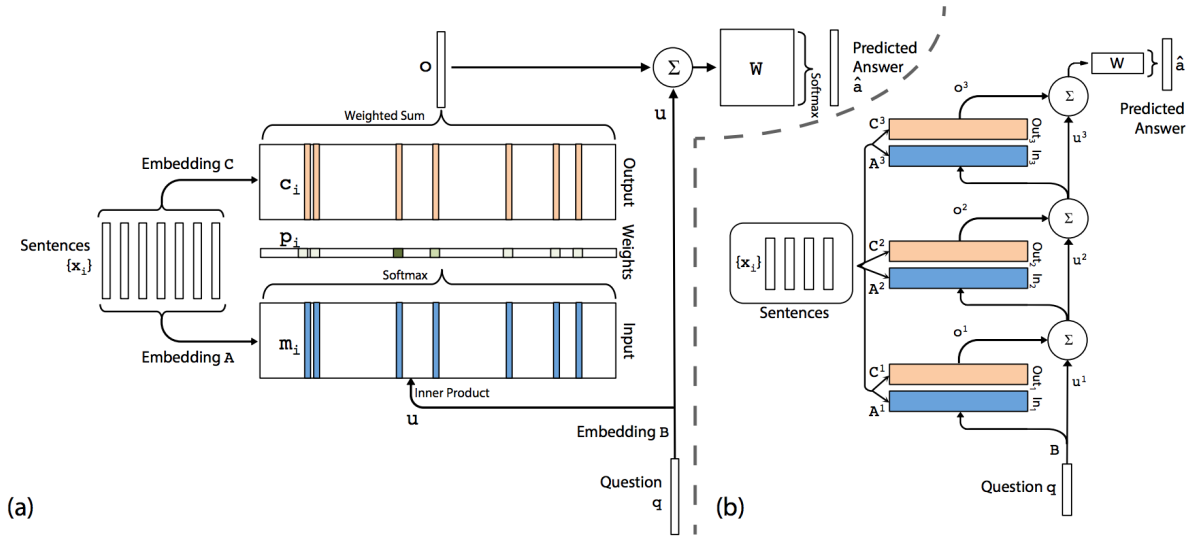


Figure 2.6: The architecture of end-to-end memory networks

In adjacent weight tying, the output embedding for one layer is the input embedding for the one above, i.e., $A^{k+1} = C^k$; meanwhile, in layer-wise weight tying, $A^k = A^{k+1}$ and $C^k = C^{k+1}$, so it is more RNN-like. In the remainder of this section, we use adjacent weight tying as the setting because it empirically outperforms the other setting.

A query vector q^k is used as a reading head. The model loops over K hops and it first computes the attention weights at hop k for each memory i using:

$$p_i^k = \text{Softmax}((q^k)^T C_i^k), \tag{2.19}$$

where C_i^k is the memory content in position i that is represented from the embedding matrix C^k. Here, p^k is a soft memory selector that decides the memory relevance with respect to the query vector q^k. The model reads out the memory o^k by the weighted-sum over C^{k+1} (using C^{k+1} from adjacent weighted tying),

$$o^k = \sum_i p_i^k C_i^{k+1}. \tag{2.20}$$

Then the query vector is updated for the next hop using

$$q^{k+1} = q^k + o^k. \tag{2.21}$$

Another essential part is what to allocate in the memory. In [12] and [39], each memory slot is represented as a sentence either from the predefined facts or utterances in dialogues, and the word embeddings in the same sentence are summed to be one single embedding for the memory slot. In [39], to let the model recognize the speaker information, the authors add the speaker

embeddings to the corresponding memory slots, in order to distinguish which utterances are from a user and which are from a system.

Story (16: basic induction)	Support	Hop 1	Hop 2	Hop 3
Brian is a frog.	yes	0.00	0.98	0.00
Lily is gray.		0.07	0.00	0.00
Brian is yellow.	yes	0.07	0.00	1.00
Julius is green.		0.06	0.00	0.00
Greg is a frog.	yes	0.76	0.02	0.00
What color is Greg? Answer: yellow	**Prediction: yellow**			

Figure 2.7: Example prediction on the simulated question-answering tasks.

This processing can be repeated several times, which is usually called multiple hops reasoning. It has been empirically proven that multiple hops are useful in several question-answering tasks. For example, in the multi-hop prediction example in Figure 2.7 using the bAbI dataset [64], there are five sentences represented as memory, and the query vector (in this case a question) is "*What color is Greg?*" As shown in the attention weights, in the first hop, the model focuses on the memory slot of "*Greg is a frog.*" After memory readout from hop one, the model pays attention to "*Brian is a frog.*" In the end, the model is able to predict that Greg's color is yellow because of the attention on "*Brian is yellow*," at the third hop.

Chapter 3

Copy-Augmented Dialogue State Tracking

In this chapter, we focus on improving the core of pipeline dialogue systems, i.e., dialogue state tracking. To effectively track the states, the model needs to memorize long dialogue context and be able to detect whether there is any slot is triggered, also what are its corresponding values. Traditionally, state tracking approaches based on the assumption that ontology is defined in advance, where all slots and their values are known. Having a predefined ontology can simplify DST into a classification problem and improve performance. However, there are two major drawbacks to this approach: 1) A full ontology is hard to obtain in advance [24]. In the industry, databases are usually accessed through an external API only, which is owned and maintained by others. It is not feasible to gain access to enumerate all the possible values for each slot. 2) Even if a full ontology exists, the number of possible slot values could be large and variable. For example, a restaurant name or a train departure time can contain a large number of possible values. Therefore, many of the previous work [21, 22, 25, 26] that are based on neural classification models may not be applicable in a real scenario.

The copy mechanism, therefore, could be essential in dialogue state tracking via copying slot values from a dialogue history to extracted states. A dialogue state tracker with copy ability can detect unknown slot values in an ontology. Here, we propose a dialogue state tracker, the **tra**nsferable **d**ialogue stat**e** generator (TRADE), which is a novel end-to-end architecture without SLU module to perform state tracking based on generative models [65]. It includes an utterance encoder, a slot gate, and a state generator. It leverages its context-enhanced slot gate and copy mechanism to properly track slot values mentioned anywhere in a dialogue history.

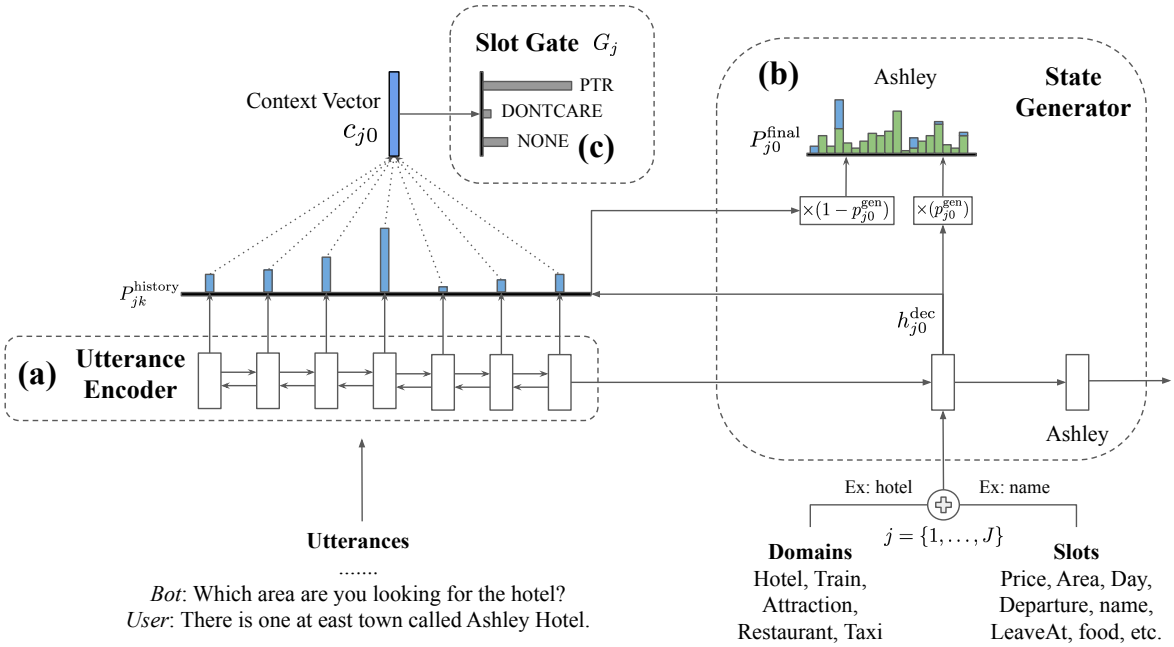


Figure 3.1: The architecture of the proposed TRADE model, which includes (a) an utterance encoder, (b) a state generator, and (c) a slot gate, all of which are shared among domains.

3.1 Model Description

The proposed TRADE model [1] in Fig. 3.1 comprises three components: an utterance encoder, a slot gate, and a state generator. Instead of predicting the probability of every predefined ontology term, this model directly generates slot values using the sequence decoding strategy. Similar to [66] for multilingual neural machine translation, we share all the model parameters, and the state generator starts with a different start-of-sentence token for each *(domain, slot)* pair.

3.1.1 Architecture

The (a) utterance encoder encodes dialogue utterances into a sequence of fixed-length vectors. The (b) state generator decodes multiple output tokens for all *(domain, slot)* pairs independently to predict their corresponding values. To determine whether any of the *(domain, slot)* pairs are mentioned, the context-enhanced (c) slot gate is used with the state generator. The context-enhanced slot gate predicts whether each of the pairs is actually triggered by the dialogue via a three-way classifier. Assuming that there are J possible *(domain, slot)* pairs in the setting.

[1] The code is available at `https://github.com/jasonwu0731/trade-dst`

(a) Utterance Encoder

Note that the utterance encoder can be any existing encoding model. We use s bi-directional GRU to encode the dialogue history. The input to the utterance encoder is the concatenation of all words in the dialogue history, and the model infers the states across a sequence of turns. We use the dialogue history as the input of the utterance encoder, rather than the current utterance only.

(b) State Generator

To generate slot values using text from the input source, a copy mechanism is required. We employ soft-gated pointer-generator copying to combine a distribution over the vocabulary and distribution over the dialogue history into a single output distribution. We use a GRU as the decoder of the state generator to predict the value for each *(domain, slot)* pair, as shown in Fig. 3.1. The state generator decodes J pairs independently. We simply supply the summed embedding of the domain and slot as the first input to the decoder.

At decoding step k for the j-th *(domain, slot)* pair, the generator GRU takes a word embedding w_{jk} as its input and returns a hidden state h_{jk}^{dec}. The state generator first maps the hidden state h_{jk}^{dec} into the vocabulary space P_{jk}^{vocab} using the trainable embedding $E \in \mathbb{R}^{|V| \times d_{hdd}}$, where $|V|$ is the vocabulary size and d_{hdd} is the hidden size. At the same time, the h_{jk}^{dec} is used to compute the history attention P_{jk}^{history} over the encoded dialogue history H_t:

$$P_{jk}^{\text{vocab}} = \text{Softmax}(E(h_{jk}^{\text{dec}})^\top) \in \mathbb{R}^{|V|}, \tag{3.1}$$

$$P_{jk}^{\text{history}} = \text{Softmax}(H_t(h_{jk}^{\text{dec}})^\top) \in \mathbb{R}^{|X_t|}. \tag{3.2}$$

The final output distribution P_{jk}^{final} is the weighted-sum of two distributions,

$$P_{jk}^{\text{final}} = p_{jk}^{\text{gen}} \times P_{jk}^{\text{vocab}} + (1 - p_{jk}^{\text{gen}}) \times P_{jk}^{\text{history}} \in \mathbb{R}^{|V|}. \tag{3.3}$$

The scalar p_{jk}^{gen} is trainable to combine the two distributions, which is computed by

$$p_{jk}^{\text{gen}} = \text{Sigmoid}(W_1[h_{jk}^{\text{dec}}; w_{jk}; c_{jk}]) \in \mathbb{R}^1, \tag{3.4}$$

$$c_{jk} = P_{jk}^{\text{history}} H_t \in \mathbb{R}^{d_{hdd}}. \tag{3.5}$$

The soft-gate copy mechanism is the same as Eq. (2.18), but here we repeat it J times to get different distributions for every *(domain, slot)* pair.

(c) Slot Gate

The context-enhanced slot gate G is a simple three-way classifier that maps a context vector taken from the encoder hidden states H_t to a probability distribution over *ptr*, *none*, and *dontcare* classes. For each *(domain, slot)* pair, if the slot gate predicts *none* or *dontcare*, we ignore the values generated by the decoder and fill the pair as "not-mentioned" or "does not care". Otherwise, we take the generated words from our state generator as its value. With a linear layer parameterized by $W_g \in \mathbb{R}^{3\times d_{hdd}}$, the slot gate for the j-th *(domain, slot)* pair is defined as

$$G_j = \text{Softmax}(W_g \cdot (c_{j0})^\top) \in \mathbb{R}^3, \tag{3.6}$$

where c_{j0} is the context vector computed in Eq (3.5) using the first decoder hidden state.

3.1.2 Optimization

During training, we optimize for both the slot gate and the state generator. For the former, the cross-entropy loss L_g is computed between the predicted slot gate G_j and the true one-hot label y_j^{gate},

$$L_g = \sum_{j=1}^{J} -\log(G_j \cdot (y_j^{\text{gate}})^\top). \tag{3.7}$$

For the latter, another cross-entropy loss L_v between P_{jk}^{final} and the true words Y_j^{label} is used. We define L_v as

$$L_v = \sum_{j=1}^{J} \sum_{k=1}^{|Y_j|} -\log(P_{jk}^{\text{final}} \cdot (y_{jk}^{\text{value}})^\top). \tag{3.8}$$

L_v is the sum of losses from all the *(domain, slot)* pairs and their decoding time steps. We optimize the weighted-sum of these two loss functions using hyper-parameters α and β,

$$L = \alpha L_g + \beta L_v. \tag{3.9}$$

3.2 Multiple Domain DST

In a single-task multi-domain dialogue setting, as shown in Fig. 3.2, a user can start a conversation by asking to reserve a restaurant, then request information regarding an attraction nearby, and finally ask to book a taxi. In this case, the DST model has to determine the corresponding domain, slot, and value at each turn of dialogue, which contains a large number of combinations in the ontology. For example, single-domain DST problems usually have only a few slots that

need to be tracked, four slots in WOZ [38] and eight slots in DSTC2 [67], but there are 30 *(domain, slot)* pairs and over 4,500 possible slot values in MultiWOZ [68], a multi-domain dialogue dataset. Another challenge in the multi-domain setting comes from the need to perform multi-turn mapping. Single-turn mapping refers to the scenario where the *(domain, slot, value)* triplet can be inferred from a single turn (the solid line in the figure), while in multi-turn mapping, it may need to be inferred from multiple turns which happen in different domains (the dotted line in the figure). For instance, the *(area, centre)* pair from the *attraction* domain in Fig. 3.2 can be predicted from the *area* information in the *restaurant* domain, which is mentioned in the preceding turns.

To tackle these challenges, we emphasize that DST models should share tracking knowledge across domains. There are many slots among different domains that share all or some of their values. For example, the *area* slot can exist in many domains, e.g., *restaurant*, *attraction*, and *taxi*. Moreover, the *name* slot in the *restaurant* domain can share the same value with the *departure* slot in the *taxi* domain. Additionally, to enable the DST model to track slots in unseen domains, transferring knowledge across multiple domains is imperative. We expect DST models can learn to track some slots in zero-shot domains by learning to track the same slots in other domains. For example, if the model learns how to track the "departure" slot in the *bus* domain, then it could transfer the knowledge to track the same slot in *taxi* domain.

3.2.1 Experimental Setup

Dataset

Multi-domain Wizard-of-Oz (MultiWOZ) is the largest existing human-human conversational corpus spanning over seven domains, containing 8438 multi-turn dialogues, with each dialogue averaging 13.68 turns. Different from existing standard datasets like WOZ [38] and DSTC2 [67], which contain less than 10 slots and only a few hundred values, MultiWOZ has 30 *(domain, slot)* pairs and over 4,500 possible values. We use the DST labels from the original training, validation and testing dataset. Only five domains (*restaurant*, *hotel*, *attraction*, *taxi*, *train*) are used in our experiment because the other two domains (*hospital*, *police*) have very few dialogues (10% compared to others) and only appear in the training set. The slots in each domain and the corresponding data size are reported in Table 3.1.

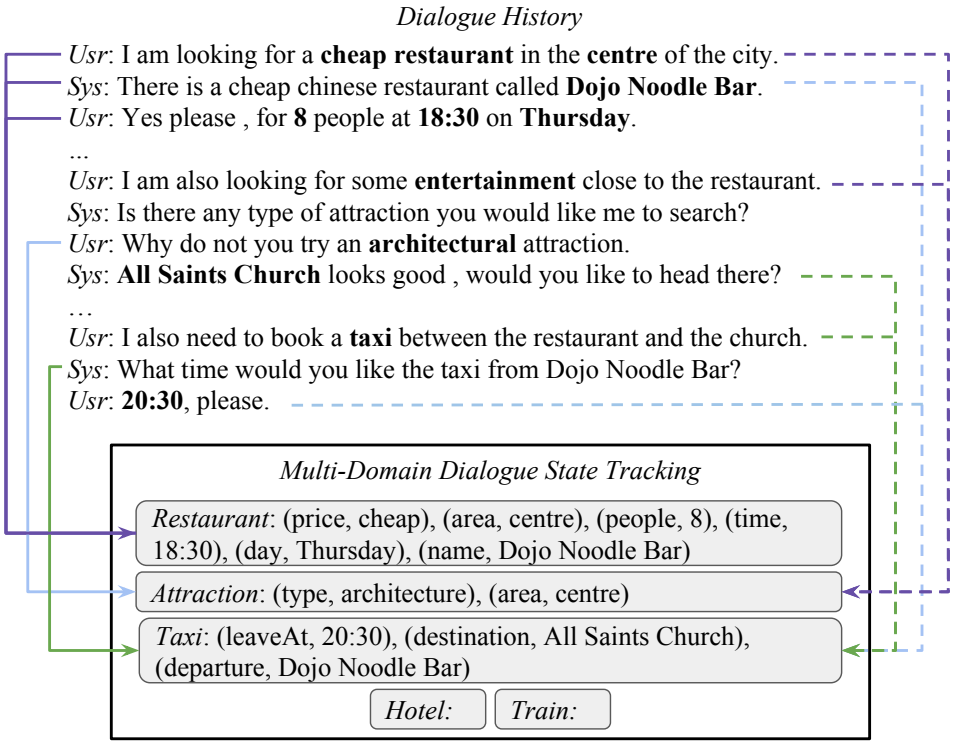


Figure 3.2: An example of multi-domain dialogue state tracking in a conversation. The solid arrows on the left are the single-turn mapping, and the dot arrows on the right are multi-turn mapping. The state tracker needs to track slot values mentioned by the user for all the slots in all the domains.

Training

The model is trained end-to-end using the Adam optimizer [69] with a batch size of 32. The learning rate annealing is in the range of $[0.001, 0.0001]$ with a dropout ratio of 0.2. Both α and β in Eq (3.9) are set to one. All the embeddings are initialized by concatenating Glove embeddings [70] and character embeddings [71], where the dimension is 400 for each vocabulary word. A greedy search decoding strategy is used for our state generator since the generated slot values are usually short in length and contain simple grammar. In addition, to increase model generalization and simulate an out-of-vocabulary setting, a word dropout is utilized with the utterance encoder by randomly masking a small number of input tokens.

Evaluation Metrics

Two evaluation metrics, joint goal accuracy and slot accuracy, are used to evaluate the performance on multi-domain DST. The joint goal accuracy compares the predicted dialogue states to the ground truth at each dialogue turn, and the output is considered correct if and only if all the predicted values exactly match the ground truth values. The slot accuracy, on the other hand,

Table 3.1: The dataset information of MultiWOZ on five different domains: hotel, train, attraction, restaurant, and taxi.

	Hotel	**Train**	**Attraction**	**Restaurant**	**Taxi**
Slots	price, type, parking, stay, day, people, area, stars, internet, name	destination, departure, day, arrive by, leave at, people	area, name, type	food, price, area, name, time, day, people	destination, departure, arrive by, leave by
Train	3381	3103	2717	3813	1654
Valid	416	484	401	438	207
Test	394	494	395	437	195

individually compares each (domain, slot, value) triplet to its ground truth label.

3.2.2 Baseline Models

We make a comparison with the following existing models: MDBT [25], GLAD [26], GCE [72], and SpanPtr [24], and we briefly describe these baseline models below:

- MDBT: A model that jointly identifies the domain and tracks the belief states corresponding to that domain is proposed. Multiple bi-LSTMs are used to encode system and user utterances. The semantic similarity between utterances and every predefined ontology term is computed separately. Each ontology term is triggered if the predicted score is greater than a threshold. The model was tested on multi-domain dialogues.

- GLAD: This model uses self-attentive RNNs to learn a global tracker that shares data among slots and a local tracker that tracks each slot. The model takes previous system actions and the current user utterance as input and computes semantic similarity with predefined ontology terms. The authors show that it can generalize on rare slot-value pairs with few training examples. The model was tested on single-domain dialogues.

- GCE: This is the current state-of-the-art model on the single-domain WOZ dataset [38] and DSTC2 [67]; it is a simplified and sped up version of GLAD, without the slot-specific

RNNs that compute for each utterance. It reduces the latency in training and inference times by 35% on average, while preserving performance of state tracking. The model was tested on single-domain dialogues.

- SpanPtr: This is the first model that applies pointer networks [52] to the DST problem, which generates both start and end pointers to perform index-based copying. The authors claim that their model based on the pointer network can effectively extract unknown slot values while still obtains promising accuracy on the DSTC2 [67] benchmark. The model was tested on single-domain dialogues.

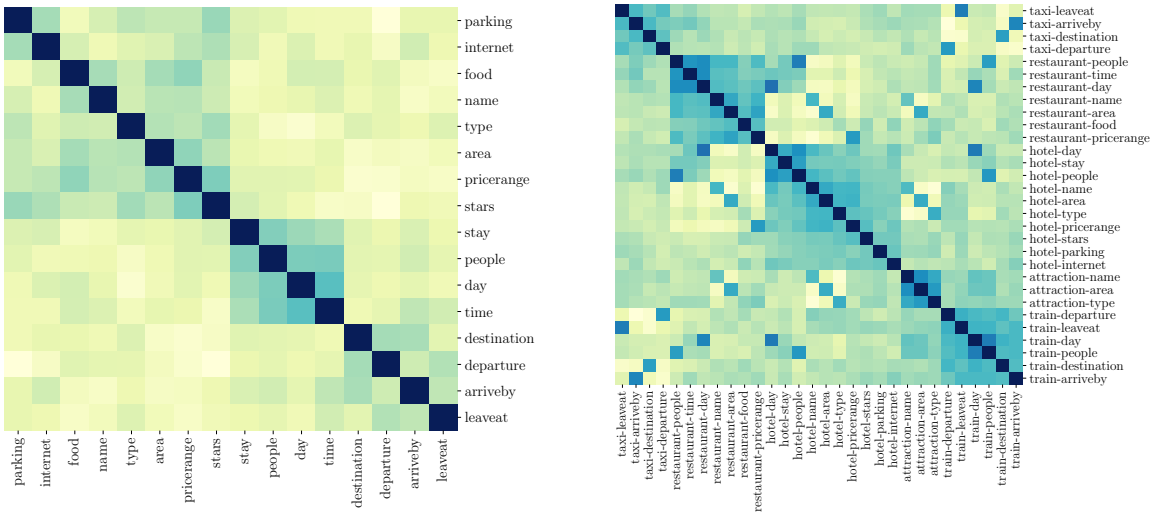
Table 3.2: The multi-domain DST evaluation on MultiWOZ and its single *restaurant* domain.

	MultiWOZ		**MultiWOZ (Only Restaurant)**	
	Joint	***Slot***	***Joint***	***Slot***
MDBT	15.57	89.53	17.98	54.99
GLAD	35.57	95.44	53.23	96.54
GCE	36.27	98.42	60.93	95.85
SpanPtr	30.28	93.85	49.12	87.89
TRADE	**48.62**	96.92	**65.35**	93.28

3.2.3 Results and Discussion

As shown in Table 3.2, TRADE achieves the highest performance, 48.62% on joint goal accuracy and 96.92% on slot accuracy, on multiWOZ. For comparison with the performance on a single domain, the results on the *restaurant* domain of MultiWOZ are reported as well. The performance difference between SpanPtr and our model mainly comes from the limitation of index-based copying. For examples, if the true label for the price range slot is *cheap*, the relevant user utterance describing the restaurant may actually be, for example, *economical*, *inexpensive*, or *cheaply*. Note that the MDBT, GLAD, and GCE models each need a predefined domain ontology to perform binary classification for each ontology term, which hinders their DST tracking performance.

We visualize the cosine similarity matrix for all the possible slot embeddings in Figure 3.3 (a). Most of the slot embeddings are not close to each other, which is expected because the model only depends on these features as start-of-sentence embeddings to distinguish different



(a) (b)

Figure 3.3: The embeddings cosine similarity visualization for (a) slots and (b) *(domain, slot)* pairs. Slots that share similar values or have correlated values learn similar embeddings. For example *destination vs. departure* (which share similar values) or *price range vs. stars* exhibit high correlation.

slots. Note that some slots are relatively close because either the values they track may share similar semantic meanings or the slots are correlated. For example, *destination* and *departure* track names of cities, while *people* and *stay* track numbers. On the other hand, *price range* and *star* in the hotel domain are correlated because high-star hotels are usually expensive. The visualization of all the possible *(domain, slot)* embeddings is shown in Figure 3.3(b).

An error analysis of multi-domain training is shown in Figure 3.4. Not surprisingly, *name* slots in the *restaurant*, *attraction*, and *hotel* domains have the highest error rates, 8.50%, 8.17%, and 7.86%, respectively. It is because this slot usually has a large number of possible values that is hard to recognize. On the other hand, number-related slots such as *arrive_by*, *people*, and *stay* usually have the lowest error rates. We also find that the *type* slot of *hotel* domain has a high error rate, even if it is an easy task with only two possible values in the ontology. The reason is that labels of the *(hotel, type)* pair are sometimes missing in the dataset, which makes our prediction incorrect even if it is supposed to be predicted.

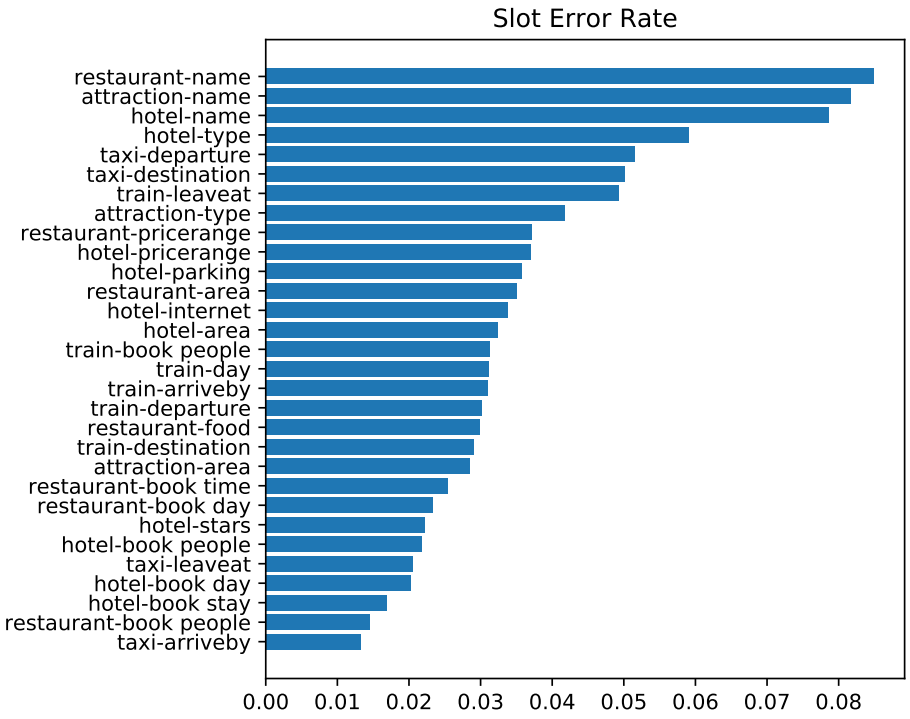


Figure 3.4: Slots error rate on test set of multi-domain training. The *name* slot in *restaurant* domain has the highest error rate, 8.50%, and the *arrive_by* slot in *taxi* domain has the lowest error rate, 1.33%

3.3 Unseen Domain DST

In this section, we focus on the ability of TRADE to generalize to an unseen domain by considering zero-shot transferring and few-shot domain expansion. In the zero-shot setting, we assume we have no training data in the new domain, while in the few-shot case, we assume just 1% of the original training data in the unseen domain is available (around 20 to 30 dialogues). One of the motivations to perform unseen domain DST is because collecting a large-scale task-oriented dataset for a new domain is expensive and time-consuming [68], and there are a large number of domains in realistic scenarios.

3.3.1 Zero-shot DST

Zero-shot applications in dialogue learning have been used for intention classifiers [17], slot-filling [73], dialogue policy [74], and language generation [66, 75]. In the DST task, ideally based on the slots already learned, a model is able to directly track those slots that are present in a new domain. For example, if the model is able to track the *departure* slot in the *train* domain,

then that ability may transfer to the *taxi* domain, which uses similar slots. Note that generative DST models take the dialogue context/history X, the domain D, and the slot S as input and then generate the corresponding values Y^{value}. Let $(X, D_{\text{source}}, S_{\text{source}}, Y^{\text{value}}_{\text{source}})$ be the set of samples seen during the training phase and $(X, D_{\text{target}}, S_{\text{target}}, Y^{\text{value}}_{\text{target}})$ the samples which the model was not trained to track. A zero-shot DST model should be able to generate the correct values of $Y^{\text{value}}_{\text{target}}$ given the context X, domain D_{target}, and slot S_{target}, without using any training samples. The same context X may appear in both source and target domains but the pairs $(D_{\text{target}}, S_{\text{target}})$ are unseen. This setting is extremely challenging if no slot in S_{target} appears in S_{source}, since the model has never been trained to track such a slot.

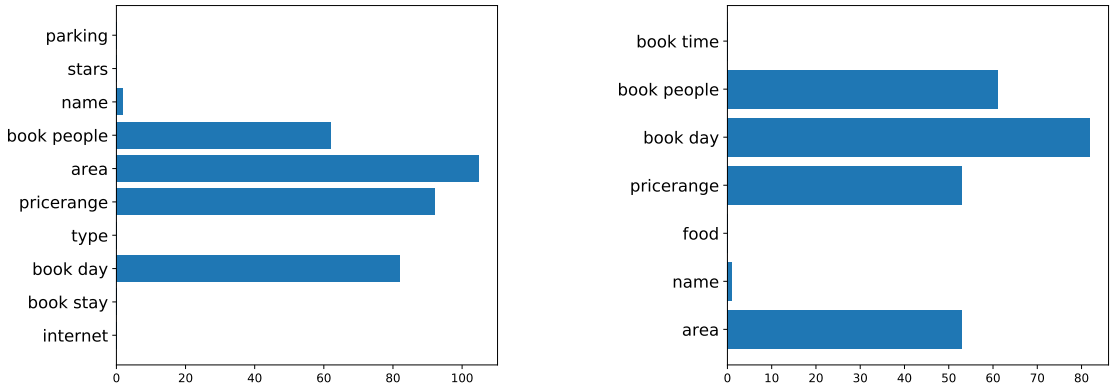
Results and Discussion

Table 3.3: Zero-shot experiments on an unseen domain. We held-out one domain each time to simulate the setting.

	Trained Single		**Zero-Shot**	
	Joint	*Slot*	*Joint*	*Slot*
Hotel	55.52	92.66	13.70	65.32
Train	77.71	95.30	22.37	49.31
Attraction	71.64	88.97	19.87	55.53
Restaurant	65.35	93.28	11.52	53.43
Taxi	76.13	89.53	**60.58**	73.92

We run zero-shot experiments by excluding one domain from the training set. As shown in Table 3.3, the *taxi* domain achieves the highest zero-shot performance, 60.58% on joint goal accuracy, which is close to the result achieved by training on all the *taxi* domain data (76.13%). Although performances on the other zero-shot domains are not especially promising, they still achieve around 50 – 65% slot accuracy without using any in-domain samples. The reason why the zero-shot performance on the *taxi* domain is high is that all four slots share similar values with the corresponding slots in the *train* domain.

In Fig. 3.5, the zero-shot analysis of two selected domains, *hotel* and *restaurant*, which contain more slots to be tracked, are shown. To better understand the behavior of knowledge transferring, here we only consider labels that are not empty, i.e., we ignore data that is labeled as "none", because predicting "none" is relatively easier for the model. In both *hotel* and *restaurant* domains, knowledge about *people, area, price_range*, and *day* slots are successfully



(a) (b)

Figure 3.5: Zero-shot DST error analysis on (a) *hotel* and (b) *restaurant* domains. The x-axis represents the number of each slot which has correct non-empty values. In *hotel* domain, the knowledge to track *people, area, price_range*, and *day* slots are successfully transferred from other domains seen in training.

transferred from the other four domains. For unseen slots that only appear in one domain, it is very hard for our model to track them correctly. For example, *parking, stars* and *internet* slots only appear in *hotel* domain, and *food* slot is unique to the *restaurant* domain.

3.3.2 Expanding DST for Few-shot Domain

In this section, we assume that only a small number of samples from the new domain $(X, D_{\text{target}}, S_{\text{target}}, Y_{\text{target}}^{\text{value}})$ are available, and the purpose is 1) to evaluate the ability of our DST model to transfer its learned knowledge to the new domain without forgetting previously learned domains; and 2) to expand the DST model to an unseen domain without expensive cost. There are two advantages to performing few-shot domain expansion: 1) being able to quickly adapt to new domains and obtain decent performance with only a small amount of training data; 2) not requiring retraining with all the data from previously learned domains, since the data may no longer be available and retraining is often very time-consuming.

Firstly, we consider a straightforward naive baseline, i.e., fine-tuning on the target domain with no constraints. Then, we employ two specific continual learning techniques: elastic weight consolidation (EWC) [76] and gradient episodic memory (GEM) [77] to fine-tune our model. We define Θ_S as the model's parameters trained in the source domain, and Θ indicates the current optimized parameters according to the target domain data.

EWC uses the diagonal of the Fisher information matrix F as a regularizer for adapting to the target domain data. This matrix is approximated using samples from the source domain.

The EWC loss is defined as

$$L_{ewc}(\Theta) = L(\Theta) + \sum_i \frac{\lambda}{2} F_i(\Theta_i - \Theta_{S,i})^2, \tag{3.10}$$

where λ is a hyper-parameter. Different from EWC, GEM keeps a small number of samples K from the source domains, and, while the model learns the new target domain, a constraint is applied on the gradient to prevent the loss on the stored samples from increasing. The training process is defined as:

$$\begin{aligned} &\text{Minimize}_{\Theta}\ L(\Theta) \\ &\text{Subject to } L(\Theta, K) \leq L(\Theta_S, K), \end{aligned} \tag{3.11}$$

where $L(\Theta, K)$ is the loss value of the K stored samples. [77] show how to solve the optimization problem in Eq (3.11) with quadratic programming if the loss of the stored samples increases.

Experimental Setup

We follow the same procedure as in the joint training section, and we run a small grid search for all the methods using the validation set. For EWC, we set different values of λ for all the domains, and the optimal value is selected using the validation set. Finally, in GEM, we set the memory size K to 1% of the source domains.

Results and Discussion

In this setting, the TRADE model is pre-trained on four domains and a *withheld* domain is reserved for domain expansion to perform fine-tuning. After fine-tuning on the new domain, we evaluate the performance of TRADE on 1) the four pre-trained domains, and 2) the new domain. We experiment with different fine-tuning strategies. In Table 3.4, the first row is the base model that is trained on the four domains. The second row is the results on the four domains after fine-tuning on 1% new domain data using three different strategies. One can find that GEM outperforms naive and EWC fine-tuning in terms of catastrophic forgetting on the four domains. Then we evaluate the results on a new domain for two cases: training from scratch and fine-tuning from the base model. Results show that fine-tuning from the base model usually achieves better results on the new domain compared to training from scratch. In general, GEM outperforms naive and EWC fine-tuning by far in terms of overcoming catastrophic forgetting.

Evaluation on 4 Domains		**Joint** *Except Hotel*	**Slot** *Except Hotel*	**Joint** *Except Train*	**Slot** *Except Train*	**Joint** *Except Attraction*	**Slot** *Except Attraction*	**Joint** *Except Restaurant*	**Slot** *Except Restaurant*	**Joint** *Except Taxi*	**Slot** *Except Taxi*
Base Model (BM) training on 4 domains		58.98	96.75	55.26	96.76	55.02	97.03	54.69	96.64	49.87	96.77
Fine-tuning BM on 1% new domain	*Naive*	36.08	93.48	23.25	90.32	40.05	95.54	32.85	91.69	46.10	96.34
	EWC	40.82	94.16	28.02	91.49	45.37	84.94	34.45	92.53	**46.88**	96.44
	GEM	**53.54**	**96.27**	**50.69**	**96.42**	**50.51**	**96.66**	**45.91**	**95.58**	46.43	**96.45**
Evaluation on New Domain		*Hotel*		*Train*		*Attraction*		*Restaurant*		*Taxi*	
Training 1% New Domain		19.53	77.33	44.24	85.66	**35.88**	**68.60**	32.72	82.39	60.38	72.82
Fine-tuning BM on 1% new domain	*Naive*	19.13	75.22	**59.83**	**90.63**	29.39	60.73	**42.42**	**86.82**	**63.81**	**79.81**
	EWC	19.35	76.25	58.10	90.33	32.28	62.43	40.93	85.80	63.61	79.65
	GEM	**19.73**	**77.92**	54.31	89.55	34.73	64.37	39.24	86.05	63.16	79.27

Table 3.4: Domain expanding DST for different few-shot domains.

We also find that pre-training followed by fine-tuning outperforms training from scratch on the single domain.

Fine-tuning TRADE with GEM maintains higher performance on the original four domains. Take the *hotel* domain as an example, the performance on the four domains after fine-tuning with GEM only drops from 58.98% to 53.54% (-5.44%) on joint accuracy, whereas naive fine-tuning deteriorates the tracking ability, dropping joint goal accuracy to 36.08% (-22.9%). Expanding TRADE from four domains to a new domain achieves better performance than training from scratch on the new domain. This observation underscores the advantages of transfer learning with the proposed TRADE model. For example, our TRADE model achieves 59.83% joint accuracy after fine-tuning using only 1% of *Train* domain data, outperforming training the*Train* domain from scratch, which achieves 44.24% using the same amount of new-domain data.

Finally, when considering *hotel* and *attraction* as a new domain, fine-tuning with GEM outperforms the naive fine-tuning approach on the new domain. To elaborate, GEM obtains 34.73% joint accuracy on the *attraction* domain, but naive fine-tuning on that domain can only achieve 29.39%. This implies that in some cases learning to keep the tracking ability (learned parameters) of the learned domains helps to achieve better performance for the new domain.

3.4 Short Summary

We introduce a transferable dialogue state generator for multi-domain dialogue state tracking, which can better memorize the long dialogue context and track the states efficiently. Our model learns to track states without any predefined domain ontology, which can handle unseen slot values using a copy mechanism. TRADE shares all of its parameters across multiple domains and achieves state-of-the-art joint goal accuracy and slot accuracy on the MultiWOZ dataset for

five different domains. Moreover, domain sharing enables TRADE to perform zero-shot DST for unseen domains. With the help of existing continual learning algorithms, our model can quickly adapt to few-shot domains without forgetting the learned ones.

Chapter 4

Retrieval-Based Memory-Augmented Dialogue Systems

In the previous chapter, we discussed how we can memorize long dialogue context via copy mechanism, and how to leverage multiple domains to further improve state tracking performance and enable unseen domain DST. In the remaining parts of this thesis, instead of solely optimizing the DST component, we view the whole dialogue system as a black box and train the system end-to-end. The inputs of the system are the long dialogue history/context and external knowledge base (KB) information, and the output is the system response for the next coming turn.

In this chapter, we first introduce one of the aspects of end-to-end dialogue learning, the retrieval-based dialogue systems. Given the dialogue history and knowledge base information, machine learning models are required to predict/select the correct system response from a predefined response candidates. This task is usually suitable for small dataset training or for dialogue systems that required regular but not diverse system behavior. We propose a delexicalization strategy to simplify the retrieval problem, then we introduce two memory-augmented neural networks, a recurrent entity networks (REN) [78] and dynamic query memory network (DQMN) [79], for task-oriented dialogue learning. Lastly, we evaluate the models on simulated bAbI dialogue [39], and also its more challenging OOV setting.

4.1 Recorded Delexicalization Copying

There are a large amount of entities in the ontology, e.g., names of restaurants. It is hard for a retrieval-based model to distinguish the minor difference in the response candidates. To simply overcome the weak entity identification problem, we propose a practical strategy, recorded

delexicalization copying (RDC), to replace each real entity value with its entity type and the order appearance in the dialogue. We also build a lookup table to record the mapping, e.g., the first user utterance in Figure 4.1, "*Book a table in Madrid for two,*" will be transformed into "*Book a table in [LOC-1] for [NUM-1].*" At the same time, *[LOC-1]* and *[NUM-1]* are stored in a lookup table as *Madrid* and *two*, respectively. Lexicalization is the reverse, copying the real entity values stored in the table to the output template. For example, when the output "*api-call [LOC-1] [NUM-1] [ATTM-1]*" is predicted, we will copy *Madrid*, *two* and *casual* to fill in the blanks. Last, we build the action template candidates by all the possible delexicalization system responses. RDC is similar to delexicalization and the entity indexing strategy in [42] and [80]. It not only decreases the learning complexity but also makes our system scalable to OOV settings.

4.2 Model Description

4.2.1 Recurrent Entity Networks

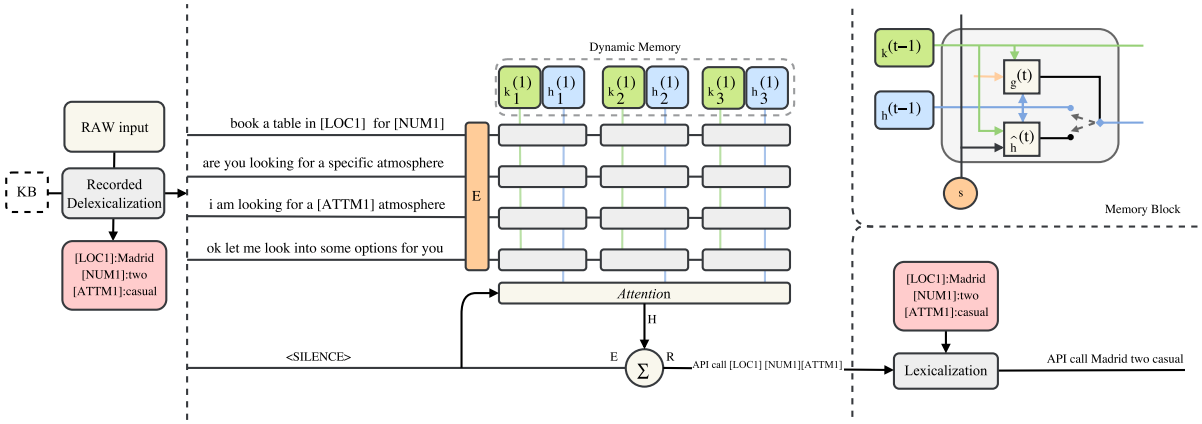


Figure 4.1: Entity-value independent recurrent entity network for goal-oriented dialogues. The graphic on the top right shows the detailed memory block.

The REN model was first used in question answering tasks and has empirically shown its effectiveness [78, 81]. It is one kind of memory-augmented neural networks that is equipped with a dynamic long-term memory, which allows it to maintain and update a representation of the state of the world as it receives new data. We first proposed to utilize REN in learning retrieval-based dialogue systems [82] in the 6th Dialogue System Technology Challenge (DSTC6) [83]. We treat every incoming utterance as the new received data, and store the dialogue history and external KB in the dynamic long-term memory to represent the state of the world.

REN has three main components: an input encoder, dynamic memory, and output module. The input encoder transforms the set of sentences s_t and the question q (we set the last user utterance as the question.) into vector representations by using multiplicative masks. We first look up word embeddings for each word in the sentences, and then apply the learned multiplicative masks, $f^{(s)}$ and $f^{(q)}$, to each word in a sentence. The final encoding vector of a sentence is defined as

$$s_t = \sum_i s_t^i \odot f_i^{(s)}, \qquad q = \sum_i q^i \odot f_i^{(q)}. \tag{4.1}$$

The dynamic memory stores long-term information, which is very similar to a GRU with a hidden state divided into blocks. The blocks ideally represent an entity type (e.g., *LOC, PRICE,* etc.), and store relevant facts about it. Each block i is made of a hidden state h_i and a key k_i. The dynamic memory module is made up of a set of blocks, which can be represented with a set of hidden states $\{h_1, \ldots, h_z\}$ and their corresponding set of keys $\{k_1, \ldots, k_z\}$. The equations used to update a generic block i are the following:

$$g_i^{(t)} = \text{Sigmoid}(s_t^\top h_i^{(t-1)} + s_t^\top k_i^{(t-1)}) \tag{4.2}$$

$$\hat{h}_i^{(t)} = \text{ReLU}(U h_i^{(t-1)} + V k_i^{(t-1)} + W s_t) \tag{4.3}$$

$$h_i^{(t)} = h_i^{(t-1)} + g_i^{(t)} \odot \hat{h}_i^{(t)} \tag{4.4}$$

$$h_i^{(t)} = h_i^{(t)} / \|h_i^{(t)}, \| \tag{4.5}$$

where $g_i^{(t)}$ is the gating function which determines how much of the ith memory should be updated, and $\hat{h}_i^{(t)}$ is the new candidate value of the memory to be combined with the existing $h_i^{(t-1)}$. The matrices U, V, and W are shared among different blocks, and are trained together with the key vectors.

The output module creates a probability distribution over the memories and hidden states using the question q. Thus, the hidden states are summed, using the probability as weight, to obtain a single vector representing all the inputs. Finally, the network output is obtained by combining the final state with the question to predict the new utterance. The model is trained using a cross-entropy loss, and it outputs the next dialogue utterance by choosing among action templates. The lexicalization step simply copies entities in the table and replaces delexicalized elements in the action template to obtain the final response.

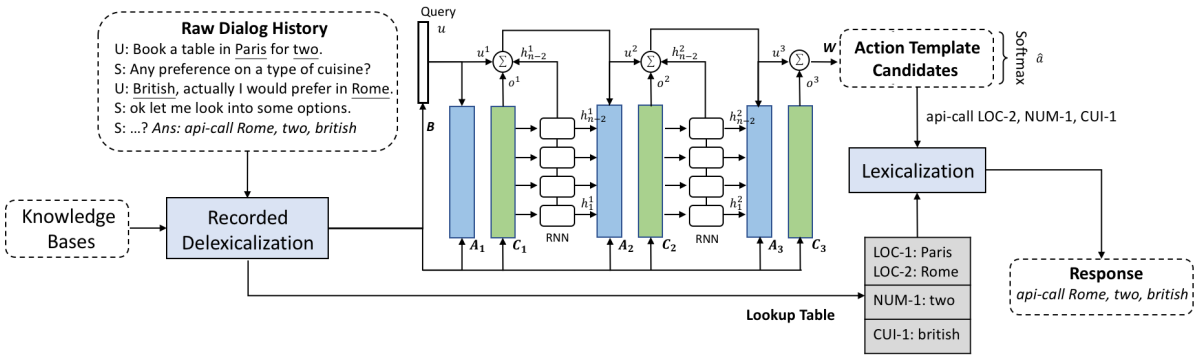


Figure 4.2: Dynamic query memory networks with recorded delexicalization copying

4.2.2 Dynamic Query Memory Networks

One major drawback of end-to-end memory networks is that they are insensitive to representing temporal dependencies between memories. To mitigate the problem, we propose a novel architecture called a dynamic query memory network (DQMN) to capture time step information in dialogues, by utilizing RNNs between memory layers to represent latent dialogue states and the dynamic query vector. We adopt the idea from [78], whose model can be seen as a bank of gated RNNs, and hidden states correspond to latent concepts and attributes. Therefore, to obtain a similar behavior, DQMN adds a recurrent architecture between memory hops in the original memory networks. We use the memory cells as the inputs of an GRU, based on the utterance order appearing in the dialogue history. The final hidden state of the GRU is added to the query u_k:

$$u^{k+1} = u^k + o^k + h_N^k, \tag{4.6}$$

where h_N^k is the last GRU hidden state at the hop k. In this way, compared to Eq. (2.21), DQMN is able to capture the global attention over memory cells o^k, and also the internal latent representation of the dialogue state h_N^k.

In addition, motivated by the query-reduction networks in [41], we use each hidden state of the corresponding time step to query the next memory cells separately. That is, the next hop query vector is not generic over all the memory cells but customized. Each cell has its unique query vector

$$q_i^{k+1} = u^{k+1} + h_i^k, \tag{4.7}$$

which is then sent to the attention computation in Eq. (2.19). DQMN considers the previous hop memory cells as a sequence of query-changing triggers, which trigger the GRU to generate more dynamically informed queries. Therefore, it can effectively alleviate temporal problems

with the dynamic query components.

4.3 Experimental Setup

4.3.1 Dataset

Table 4.1: Statistics of bAbI dialogue dataset.

Task	**1**	**2**	**3**	**4**	**5**
Avg. User turns	4	6.5	6.4	3.5	12.9
Avg. Sys turns	6	9.5	9.9	3.5	18.4
Avg. KB results	0	0	24	7	23.7
Avg. Sys words	6.3	6.2	7.2	5.7	6.5
Vocabulary	3747				
Train dialogs	1000				
Val dialogs	1000				
Test dialogs	1000 + 1000 OOV				

We use the bAbI dialogue dataset [39] to evaluate the performance. It is one of the standard benchmarks since it evaluates all the desirable features of an end-to-end task-oriented dialogue system. The dataset is divided into five different tasks, each of which has its own training, validation, and test set. Tasks 1–4 are issuing an API call, refining an API call, recommending options, and providing additional information, respectively. Task 5 (full dialogues) is a union of tasks 1–4 and includes more conversational turns.

- Task 1 Issuing API calls: A user request implicitly defines a query that can contain the required fields. The bot must ask questions for filling the missing fields and eventually generate the correct corresponding API call. The bot asks for information in a deterministic order, making prediction possible.

- Task 2 Updating API calls: Starting by issuing an API call as in Task 1, users then ask to update their requests between 1 and 4 times. The order in which fields are updated is random. The bot must ask users if they are done with their updates and issue the updated API call.

- Task 3 Displaying options: Given a user request, the KB is queried using the corresponding API call and add the facts resulting from the call to the dialogue history. The bot must propose options to users by listing the restaurant names sorted by their corresponding rating (from higher to lower) until users accept.

- Task 4 Providing extra information: Given a user request, a restaurant is sampled and start the dialogue as if users had agreed to book a table there. We add all KB facts corresponding to it to the dialogue. Users then ask for the phone number of the restaurant, its address or both. The bot must learn to use the KB facts correctly to answer.
- Task 5 Conducting full dialogues: Tasks 1–4 are combined to generate full dialogues.

There are two test sets for each task: one follows the same distribution as the training set and the other has OOV words from a different KB, i.e., the slot values that do not appear in the training set. Dataset statistics are reported in Table 4.1.

4.3.2 Training

Recurrent Entity Networks

For each task, we fix the number of the memory block to five because there are five different slots, and we use Adam [69] optimizer with learning rate 0.1 and dropout ratio 0.3. The weights are initialized randomly from a Gaussian distribution with zero mean and $\sigma = 0.1$. The gradient is clipped to a maximum of 40 to avoid gradient explosion. We try a small grid search over hyper-parameters such as batch size and embedding size. Then, the setting that achieves the highest accuracy in validation is selected for the final evaluation.

Dynamic Query Memory Network

All experiments used a $K = 3$ hops model with the adjacent weight sharing scheme, e.g., $\mathbf{C}^k = \mathbf{A}^{k+1}$ for $k = 1, 2$. For sentence representation, we use position encoding bag-of-words as in [12] to capture words order. Our model is using a learning rate of 0.01, annealing half every 25 epochs until 100 epochs are reached. The weights are initialized randomly from a Gaussian distribution with zero mean and $\sigma = 0.1$. All training uses a batch size of 16 (but the cost is not averaged over a batch), and gradients with a $L2$ norm greater than 40 are clipped.

4.3.3 Evaluation Metrics

Per-response/dialogue Accuracy Per-response accuracy is considered correct only if the predicted system response is exactly the same as the gold system response. In addition, per-dialogue accuracy takes every turn in one dialogue into account. It is considered correct if

all the system responses are exactly the same as the true dialogue. These are strict evaluation metrics and are highly dependent on the system behavior, which may only be suitable for a simulated dialogue dataset, where all the responses are regular and standardized.

4.4 Results and Discussion

4.4.1 Quantitative Results

Table 4.2: Per-response accuracy and per-dialogue accuracy (in parentheses) on bAbI dialogue dataset using REN and DQMN.

Task	**MN**	**GMN**	**REN**	**DQMN**	**REN+RDC**	**DQMN+RDC**
T1	99.9 (99.6)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T2	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T3	74.9 (2.0)	74.9 (3.0)	74.9 (0)	74.9 (2.0)	91.4 (40.3)	98.7 (90.8)
T4	59.5 (3.0)	57.2 (0)	57.2 (0)	57.2 (0)	100 (100)	100 (100)
T5	96.1 (49.4)	96.3 (52.5)	97.3 (46.3)	99.2 (88.7)	97.5 (62.5)	99.9 (98.3)
T1-OOV	72.3 (0)	82.4 (0)	81.8 (0)	82.5 (0)	100 (100)	100 (100)
T2-OOV	78.9 (0)	78.9 (0)	78.9 (0)	78.9 (0)	100 (100)	100 (100)
T3-OOV	74.4 (0)	75.3 (0)	75.3 (0)	74.9 (0)	89.6 (29.8)	98.7 (90.4)
T4-OOV	57.6 (0)	57.0 (0)	57.0 (0)	57.0 (0)	100 (100)	100 (100)
T5-OOV	65.5 (0)	66.7 (0)	65.1 (0)	72.0 (0)	96.0 (46.3)	99.4 (91.6)

In Table 4.2, we report the results obtained on the bAbI dialogue test sets (including the OOV). We compare our proposed models with and without RDC to the original end-to-end memory networks (MN) [12] and gated memory networks (GMN) [40], which are both retrieval-based models.

First, we discuss the model performance without the RDC strategy. On the full dialogue task (T5), REN and DQMN outperform memory network and GMN, and DQMN achieves the highest, 99.2% per-response accuracy and 88.7% per-dialogue accuracy. This result shows that the dynamic query components in DQMN allow the memory network to learn a more complex dialogue policy. Task 5 includes long conversational turns, and it requires a stronger dialogue state tracking ability. Although there is no performance difference between our models and the other baselines for T1 to T4, we can still observe a better generalization ability of our models on the OOV test set. For example, our model DQMN achieves 72.0% per-response accuracy in the T5-OOV setting, which is 7% better than others.

Next, we show the effectiveness of the RDC strategy by applying it to both REN and DQMN. On T3 the restaurant recommendation task, REN with RDC improves the performance

by 16.5% on per-response accuracy, and DQMN with RDC improves by 23.8%. On T4, the providing additional information task, both REN and DQMN with RDC can achieve perfect performance. On T5 full dialogue task, DQMN with RDC achieves 99.9% per-response accuracy and 98.3% per-dialogue accuracy. Note that with RDC, the DQMN model can achieve almost perfect per-response accuracy, even on Task5-OOV, which also confirms our initial assumption that using RDC strongly decreases the learning complexity. This strategy leads to an overall accuracy improvement, which is particularly useful when the network needs to learn how to work with abstract OOV entities.

4.4.2 Visualization

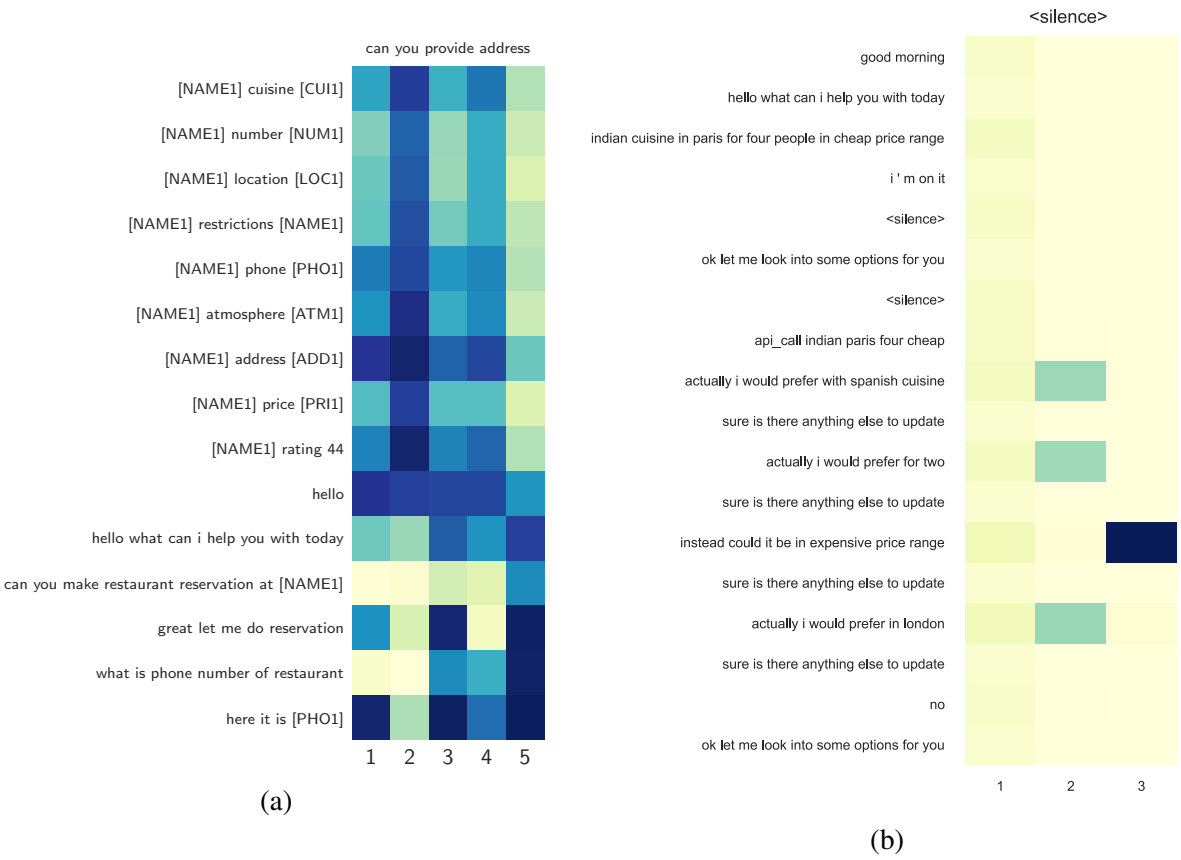


Figure 4.3: Heatmap representation of the (a) gating function for each memory block in the REN model and (b) memory attention for each hop in DQMN.

To better understand the REN behavior, we visualize the gating activation function in Figure 4.3(a). The output of this function decides how much and what we store in each memory cell. We take the model trained on T4 (i.e., providing additional information) for the visualization. We plot the activation matrix of the gate function and observe how REN learns to store

relevant information. As we can see in the figure, the model opens the memory gate once a useful information appears as input, and closes the gate for other useless sentences. Different memory blocks may focus on different information. For example, block 5 stores more information from the discourse rather than explicit KB knowledge; and block 2 opens its gate fully when the address and rating information is provided. In this case, the last user utterance (question) is "*Can you provide address?*," we can get the correct prediction because the latent address feature is represented in those memory blocks that open during the utterance "*[NAME1] address [ADD1]*".

We visualize the memory attentions of different hops in DQMN in Figure 4.3(b). One can observe that in the first hop, the model usually pays attention to almost every memory slot, which intuitively means that the model is "understanding" the general dialogue flow. In the second hop, the model focuses on three different slots (Spanish cuisine, two people, and London location) because the user has changed his/her mind to modify the intention. In the third hop, the model becomes very sharp on the utterance that is related to the price slot. In the end, DQMN gets all of the information it needs and predicts the output response "*api_call Spanish London two expensive*".

4.5 Short Summary

REN and DQMN are two memory-augmented frameworks for retrieval-based task-oriented dialogue systems, which are designed to model long dialogue context and external knowledge more efficiently. They are designed to overcome the drawbacks in retrieval-based dialogue applications that they are hard to capture long-term dependencies. A recorded delexicalization copy mechanism is utilized to reduce the learning complexity and also alleviate out-of-vocabulary entity problems. The experimental results show that our models outperform other memory networks, especially on a task with longer dialogue turns.

Chapter 5

Generation-Based Memory-Augmented Dialogue Systems

In the previous chapter, we discussed how to effectively incorporate long dialogue context and external knowledge base (KB) into retrieval-based dialogue systems. However, although they may be one of the most robust dialogue systems, they have two main drawbacks: 1) Retrieved responses are too regular and limited. While facing the real users, the system is not able to reply if they say something out-of-domain, which is not predefined in the response candidates. 2) Although using record delexicalization strategy can simplify the problem, the model does not deal with the real entity values in this case and might lose some information implied. For example, when users ask for a French restaurant, they may imply that the price of the dinner they want might not be cheap. This information is missing when the values are replaced by the slot type, i.e., replace "French" with "CUISINE-1".

In this chapter, on the other hand, we cope with another challenging approach of end-to-end dialogue learning, which is system response generation problem. Given the dialogue history and KB information, machine learning models are required to generate the system response word-by-word using recurrent structures. Compared to solely doing retrieval from the response candidates, generated responses can be more diverse, human-like, and have the potential to generalize to an unseen scenario. We propose two models, a memory-to-Sequence (Mem2Seq) model [84] and a global-to-local memory pointer network (GLMP) [85], to effectively incorporate long dialogue context and external knowledge into end-to-end learning.

A multi-turn dialogue between a driver and an agent is shown in Table 5.1. The upper part of the table is the KB information available, which includes different points-of-interest (POIs), and their corresponding addresses, types, traffic information, and distances. One can find that

Table 5.1: Multi-turn dialogue example for an in-car assistant in the navigation domain.

Distance	Traffic_info	Poi_type	Address	Poi
5 miles	moderate traffic	rest stop	329 El Camino Real	The Westin
4 miles	no traffic	pizza restaurant	113 Anton Ct	Round Table
5 miles	no traffic	chinese restaurant	271 Springer Street	Mandarin Roots
4 miles	moderate traffic	coffee or tea place	436 Alger Dr	Palo Alto Cafe
6 miles	heavy traffic	pizza restaurant	776 Arastradero Rd	Dominos
6 miles	no traffic	hospital	214 El Camino Real	Stanford Express Care
2 miles	heavy traffic	rest stop	578 Arbol Dr	Hotel Keen

1st Turn	**DRIVER**	Where can I get tea?
	System	Palo Alto Cafe is 4 miles away and serves coffee and tea. Do you want the address?
2nd Turn	**DRIVER**	Yes.
	System	Palo Alto is located at 436 Alger Dr.

the system responses include multiple entities that are existed in the table, e.g., *Palo Alto Cafe*, *4 miles*, and *436 Alger Dr*. Therefore, the ability to reason over the KB information and copy the entities from the KB to the response is essential.

5.1 Memory-to-Sequence

As mentioned in Section 2.3, existing LSTM or GRUs usually are unable to incorporate external knowledge into end-to-end learning. We present a novel architecture called Mem2Seq to learn task-oriented dialogues in an end-to-end manner. This model augments the existing memory network framework with a sequential generative architecture, using global multi-hop attention mechanisms to copy words directly from dialogue history or KBs. Mem2Seq is the first model to combine multi-hop attention mechanisms with the idea of pointer networks, which allows us to effectively incorporate KB information. It also learns how to generate dynamic queries to control memory access, and we visualize and interpret the model dynamics among hops for both the memory controller and the attention. Lastly, Mem2Seq can be trained faster and achieves state-of-the-art results on several task-oriented dialogue datasets.

5.1.1 Model Description

Mem2Seq [1] comprises of two components: an MN encoder and a memory decoder, as shown in Figure 5.1. The memory network encoder creates a vector representation of the dialogue history. Then the memory decoder reads and copies the memory to generate a response.

[1]The code is available at `https://github.com/HLTCHKUST/Mem2Seq`

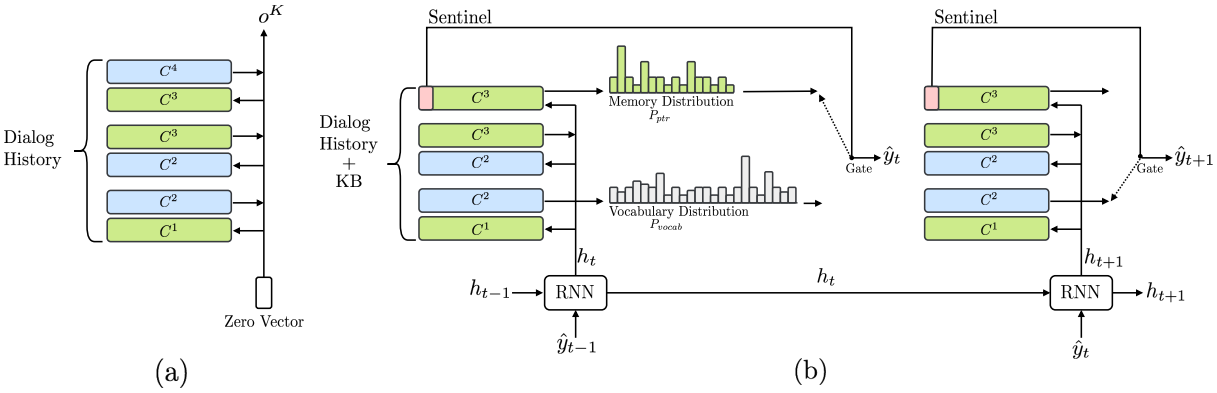


Figure 5.1: The proposed Mem2Seq architecture for task-oriented dialogue systems. (a) Memory encoder with three hops and (b) memory decoder over two-step generation.

Memory Content

We store word-level content in the memory module. Similar to [39], we add temporal information and speaker information to capture the sequential dependencies. For example, "*hello* $t1$ $\$u$" means "*hello*" at time step 1 spoken by a user. On the other hand, to store the KB information, we follow the works [86] and [87], which use a *(subject, relation, object)* representation. For example, we represent the information of *Dominos* in Table 5.1: *(Dominos, Distance, 6 miles)*. Then we sum word embeddings of the subject, relation, and object to obtain each KB memory representation. During the decoding stage, the object part is used as the generated word for copying. For instance, when the KB triplet *(Dominos, Distance, 6 miles)* is pointed to, our model copies "*6 miles*" as an output word.

Memory Encoder

Mem2Seq uses a standard memory network with adjacent weighted tying as an encoder. The input of the encoder is the dialogue history only because we believe that the encoder does not require KB information to track dialogue states. The memories of MemNN are represented by a set of trainable embedding matrices, and a query vector is used as a reading head. The model loops over K hops and it computes the attention weights at hop k for each memory. Mem2Seq will return a soft memory selector that decides the memory relevance with respect to the query vector. The model also reads out the memory by the weighted-sum of embeddings. The result from the encoding step is the memory readout vector, which will become the initial state of the decoder RNN.

Memory Decoder

The decoder uses an RNN and MN. The MN is loaded with both the dialogue history and external knowledge since we use both to generate a proper system response. A GRU is used as a dynamic query generator for the MN. At each decoding step t, the GRU gets the previously generated word and the previous query as input, and it generates the new query vector. Then the query is passed to the MN, which will produce the token. At each time step, two distributions are generated: one over all the words in the vocabulary (P_{vocab}) and the other over the memory contents (P_{ptr}). The first, P_{vocab}, is generated by concatenating the first hop memory readout and the current query vector:

$$P_{vocab}(\hat{y}_t) = \text{Softmax}(W[h_t^{dec}; o^1]). \tag{5.1}$$

On the other hand, P_{ptr} is generated using the attention weights at the last MN hop of the decoder. The decoder generates tokens by pointing to the input words in the memory, which is a similar mechanism to the attention used in pointer networks [52].

If the expected word does not appear in the memories, P_{ptr} is trained to produce the sentinel token \$. To sum up, once the sentinel is chosen, our model generates the token from P_{vocab}; otherwise, it takes the memory content using the P_{ptr} distribution. Basically, the sentinel token is used as a hard gate to control which distribution to use at each time step. A similar approach has been used in [55] to control a soft gate in a language modeling task. With this method, the model does not need to learn a gating function separately as in [53], and is not constrained by a soft gate function as in [88].

We designed our architecture in this way because we expect the attention weights in the first and the last hop to show a "looser" and "sharper" distribution, respectively. To elaborate, the first hop focuses more on retrieving memory information and the last tends to choose the exact token leveraging the pointer supervision. Hence, during training, all the parameters are jointly learned by minimizing the sum of two standard cross-entropy losses: one between P_{vocab} and the true response word for the vocabulary distribution, and one between P_{ptr} and the true memory position for the memory distribution.

Table 5.2: Dataset statistics for three different datasets, bAbI dialogue, DSTC2, and In-Car Assistant.

Task	**1**	**2**	**3**	**4**	**5**	**DSTC2**	**In-Car**
Avg. User turns	4	6.5	6.4	3.5	12.9	6.7	2.6
Avg. Sys turns	6	9.5	9.9	3.5	18.4	9.3	2.6
Avg. KB results	0	0	24	7	23.7	39.5	66.1
Avg. Sys words	6.3	6.2	7.2	5.7	6.5	10.2	8.6
Max. Sys words	9	9	9	8	9	29	87
Pointer Ratio	.23	.53	.46	.19	.60	.46	.42
Vocabulary	3747					1229	1601
Train dialogues	1000					1618	2425
Val dialogues	1000					500	302
Test dialogues	1000 + 1000 OOV					1117	304

5.1.2 Experimental Setup

Dataset

We use three public multi-turn task-oriented dialogue datasets to evaluate our model: the bAbI dialogue [39], Dialogue State Tracking Challenge 2 (DSTC2) [89] and In-Car Assistant [87]. The training/validation/test sets of these three datasets have been split in advance by the providers. The dataset statistics are reported in Table 5.2.

The bAbI dialogue dataset was introduced in Section 4.3. For the dialogues extracted from the DSTC2 we used the refined version from [39], which ignores the dialogue state annotations. The main difference from the bAbI dialogue dataset is that this dataset is extracted from real human-bot dialogues, which are noisier and harder since the bots make mistakes due to speech recognition errors or misinterpretations. The In-Car Assistant dataset, meanwhile, is a human-human, multi-domain dialogue dataset collected from Amazon Mechanical Turk. It has three distinct domains: calendar scheduling, weather information retrieval, and point-of-interest navigation. This dataset has shorter conversation turns, but the user and system behaviors are more diverse. In addition, the system responses are variant and the KB information is much more complicated. Hence, this dataset requires a stronger ability to interact with KBs, rather than dialogue state tracking.

Evaluation Metrics

Per-response/dialogue Accuracy For per-response accuracy, a generated response is correct only if it is exactly the same as the gold response; for per-dialogue accuracy, a dialogue is

considered correct only if every system response in the dialogue is exactly the same as the gold response. These evaluation metrics are only suitable for the bAbI dialogue dataset because it is a simulated dataset with very regular system behavior. Note that [39] tests their model by selecting the system response from predefined response candidates; that is, their system solves a multi-class classification problem. On the other hand, Mem2Seq generates each token individually so evaluating with these metrics is more challenging.

BLEU BLEU [90] is a measure commonly used for machine translation systems, but it has also been used in evaluating dialogue systems [44, 80] and chat-bots [91]. Moreover, the BLEU score is a relevant measure on task-oriented dialogues as there is little variance between the generated answers, unlike open-domain generation.

Entity F1 We micro-average over the entire set of system responses and compare the entities in plain text. The entities in each gold system response are selected by a predefined entity list. This metric evaluates the ability to generate relevant entities from the provided KBs and to capture the semantics of the dialogue [44, 87].

5.1.3 Results and Discussion

We compare Mem2Seq with hop 1, 3, and 6 hops with several existing models: an end-to-end memory networks (MN), gated end-to-end memory networks (GMN), and dynamic query memory networks (DQMN). We also implement the following baseline models: a sequence-to-sequence (Seq2Seq) model with and without attention [51], and pointing to unknown (Ptr-Unk) [53].

Quantitative Study

bAbI Dialogue As shown in Table 5.3, Mem2Seq with 6 hops achieves 84.5% per-response accuracy for the OOV test set, which surpasses existing methods by far. This indicates that our model can generalize well, with only 13.4% performance loss for test OOV data, while the others have around a 25–35% drop. This performance gain on the OOV data can be mainly attributed to the use of the copy mechanism. In addition, the effectiveness of the hops is demonstrated in tasks 3–5, since they require reasoning ability over the KB information. Note that the

Table 5.3: Mem2Seq evaluation on simulated bAbI dialogues. Generation methods, especially with copy mechanism, outperform other retrieval baselines.

Task	MN	GMN	DQMN	Seq2Seq	Seq2Seq+Attn	Ptr-Unk	**Mem2Seq K1**	**Mem2Seq K3**	**Mem2Seq K6**
T1	99.9 (99.6)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T2	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T3	74.9 (2.0)	74.9 (0)	74.9 (2.0)	74.8 (0)	74.8 (0)	85.1 (19.0)	87.0 (25.2)	94.5 (59.6)	**94.7 (62.1)**
T4	59.5 (3.0)	57.2 (0)	57.2 (0)	57.2 (0)	57.2 (0)	**100 (100)**	97.6 (91.7)	**100 (100)**	**100 (100)**
T5	96.1 (49.4)	96.3 (52.5)	99.2 (88.7)	98.8 (81.5)	98.4 (87.3)	**99.4 (91.5)**	96.1 (45.3)	98.2 (72.9)	97.9 (69.6)
T1-OOV	72.3 (0)	82.4 (0)	82.5 (0)	79.9 (0)	81.7 (0)	92.5 (54.7)	93.4 (60.4)	91.3 (52.0)	**94.0 (62.2)**
T2-OOV	78.9 (0)	78.9 (0)	78.9 (0)	78.9 (0)	78.9 (0)	83.2 (0)	81.7 (1.2)	84.7 (7.3)	**86.5 (12.4)**
T3-OOV	74.4 (0)	75.3 (0)	74.9 (0)	74.3 (0)	75.3 (0)	82.9 (13.4)	86.6 (26.2)	**93.2 (53.3)**	90.3 (38.7)
T4-OOV	57.6 (0)	57.0 (0)	57.0 (0)	57.0 (0)	57.0 (0)	**100 (100)**	97.3 (90.6)	**100 (100)**	**100 (100)**
T5-OOV	65.5 (0)	66.7 (0)	72.0 (0)	67.4 (0)	65.7 (0)	73.6 (0)	67.6 (0)	78.1 (0.4)	**84.5 (2.3)**

MN, GMN, and DQMN view bAbI dialogue tasks as classification problems, which are easier to solve compared to our generative methods. Finally, one can find that the Seq2Seq and Ptr-Unk models are also strong baselines, which further confirms that generative methods can achieve good performance in task-oriented dialogue systems [44].

Table 5.4: Mem2Seq evaluation on human-robot DSTC2. We make a comparison based on entity F1 score, and per-response/dialogue accuracy is low in general.

	Ent. F1	**BLEU**	**Per-Resp.**	**Per-dial.**
Rule-Based	-	-	33.3	-
QRN	-	-	43.8	-
MN	-	-	41.1	0.0
GMN	-	-	**47.4**	1.4
Seq2Seq	69.7	55.0	46.4	**1.5**
+Attn	67.1	**56.6**	46.0	1.4
+Copy	71.6	55.4	47.3	1.3
Mem2Seq K1	72.9	53.7	41.7	0.0
Mem2Seq K3	**75.3**	55.3	45.0	0.5
Mem2Seq K6	72.8	53.6	42.8	0.7

DSTC2 In Table 5.4, the results compared to the Seq2Seq models from [44] and the rule-based model from [39] are reported. Mem2Seq achieves the highest, 75.3%m BLEU score among the other baselines. Note that the per-response accuracy for every model is less than 50% since the dataset is quite noisy and it is hard to generate a response that is exactly the same as the gold response.

In-Car Assistant In Table 5.5, our model can achieve highest the highest BLEU score, or 12.6, and a 33.4% entity F1 score, which surpasses other models. Note that baselines such as Seq2Seq and Ptr-Unk have especially poor performances on this dataset since it is very inefficient for RNN methods to encode longer KB information. This is the advantage of Mem2Seq.

Furthermore, we observe the interesting phenomenon that humans can easily achieve a high

Table 5.5: Mem2Seq evaluation on human-human In-Car Assistant dataset.

	BLEU	Ent. F1	Sch. F1	Wea. F1	Nav. F1
Human	13.5	60.7	64.3	61.6	55.2
Rule-Based	6.6	43.8	61.3	39.5	40.4
Seq2Seq	8.4	10.3	09.7	14.1	07.0
+Attn	9.3	19.9	23.4	25.6	10.8
Ptr-Unk	8.3	22.7	26.9	26.7	14.9
Mem2Seq H1	11.6	32.4	39.8	**33.6**	**24.6**
Mem2Seq H3	**12.6**	**33.4**	**49.3**	32.8	20.0
Mem2Seq H6	9.9	23.6	34.3	33.0	4.4

entity F1 score with a low BLEU score. This implies that stronger reasoning ability over entities (hops) is crucial, but the results may not be similar to the golden answer. We believe humans could produce good answers even with a low BLEU score since there are different ways to express the same concept. Therefore, Mem2Seq shows the potential to successfully choose the correct entities.

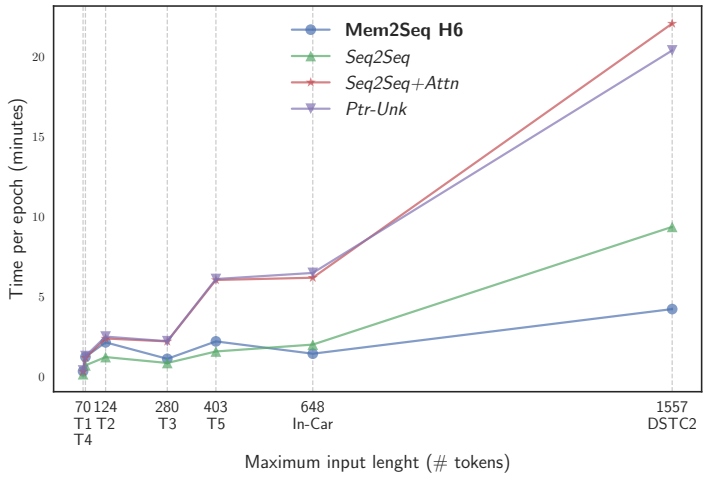


Figure 5.2: Training time per-epoch for different tasks.

Time Per-Epoch We also compare the training times, as shown in Figure 5.2, using an Intel(R) Core(TM) i7-3930K CPU@3.20GHz, with a GeForce GTX 1080 Ti. The experiments are set with batch size 16, and we report each model with the hyper-parameter that can achieve the highest performance. One can observe that the training time is not that different for short input lengths (bAbI dialogue tasks 1–4) and the gap becomes larger as the maximal input length increases. Mem2Seq is around five times faster on In-Car Assistant and DSTC2 compared to Seq2Seq with attention. This difference in training efficiency is mainly attributed to the fact that Seq2Seq models have input sequential dependencies which limit any parallelization. Moreover,

it is unavoidable for Seq2Seq models to encode KBs, while Mem2Seq only encodes with dialogue history.

Qualitative Study

Table 5.6, Table 5.7, Table 5.8 and show the generated responses of different models on the three test set samples from the In-Car Assistant dataset. Seq2Seq generally cannot produce related information, and sometimes fails in language modeling. Using attention instead helps with this issue, but it still rarely produces the correct entities. For example, Seq2Seq with attention generated "*5 miles*" but the correct response is "*4 miles*". In addition, Ptr-Unk often cannot copy the correct token from the input, as shown by "PAD". On the other hand, Mem2Seq is able to produce the correct responses in these two examples. In particular, in the navigation domain, Mem2Seq produces a different but still correct utterance.

Table 5.6: Example of generated responses for the In-Car Assistant on the navigation domain.

Distance	Traffic_info	Poi_type	Address	Poi
5 miles	moderate traffic	rest stop	329 El Camino Real	The Westin
4 miles	no traffic	pizza restaurant	113 Anton Ct	Round Table
5 miles	no traffic	chinese restaurant	271 Springer Street	Mandarin Roots
4 miles	moderate traffic	coffee or tea place	436 Alger Dr	Palo Alto Cafe
6 miles	heavy traffic	pizza restaurant	776 Arastradero Rd	Dominos
6 miles	no traffic	hospital	214 El Camino Real	Stanford Express Care
2 miles	heavy traffic	rest stop	578 Arbol Dr	Hotel Keen

1th Turn	**DRIVER**	Where can I get tea?
	Seq2Seq	I have a away from away would you like the address
	+Attn	I have a listing for a place that serves tea that is 5 miles away
	Ptr-Unk	There is a away you like would you like more info
	Mem2Seq	The nearest tea shop is Palo Alto Cafe located 4 miles away, would you like directions there?
	GOLD	Palo Alto Cafe is 4 miles away and serves coffee and tea. Do you want the address?
2th Turn	**DRIVER**	Yes.
	Seq2Seq	The is at at
	+Attn	The address is 329 El Camino Real and it's 3 miles away and there is no traffic
	Ptr-Unk	Palo Alto Cafe is 4 miles away and PAD is no traffic
	Mem2Seq	Palo Alto Cafe is 4 miles away at 436 Alger Drive
	GOLD	Palo Alto is located at 436 Alger Dr.

Table 5.7: Example of generated responses for the In-Car Assistant on the scheduling domain.

room	time	date	party	event
conference room 100	3pm	Thursday	sales team	conference
-	11am	Tuesday	-	tennis activity
-	7pm	Wednesday	-	dentist appointment
conference room 100	1pm	Tuesday	HR	meeting
-	7pm	Friday	-	doctor appointment
-	11am	Tuesday	-	lab appointment

DRIVER	When's my tennis match?
Seq2seq	Your lab appointment is on Monday at
+Att	Your lab appointment is on Tuesday at 11am
Ptr-Unk	Your tennis match is on PAD at 1pm
Mem2Seq	Your tennis is on Tuesday at 11am
GOLD	Your tennis match is Tuesday at 11am

Table 5.8: Example of generated responses for the In-Car Assistant on the weather domain.

location	monday	tuesday	wednesday
grand rapids	hot, low of 50F, high of 70F	raining, low of 60F, high of 80F	rain, low of 20F, high of 30F
new york	misty, low of 30F, high of 50F	snow, low of 70F, high of 80F	cloudy, low of 20F, high of 30F
boston	hail, low of 90F, high of 100F	overcast, low of 60F, high of 70F	rain, low of 50F, high of 60F
durham	hot, low of 90F, high of 100F	dry, low of 60F, high of 80F	misty, low of 60F, high of 80F
san francisco	rain, low of 60F, high of 70F	cloudy, low of 30F, high of 40F	overcast, low of 90F, high of 100F
carson	raining, low of 70F, high of 80F	humid, low of 90F, high of 100F	frost, low of 40F, high of 60F
san jose	blizzard, low of 40F, high of 50F	snow, low of 90F, high of 100F	overcast, low of 60F, high of 80F
thursday	**friday**	**saturday**	**sunday**
clear skies, low of 60F, high of 70F	warm, low of 70F, high of 90F	foggy, low of 50F, high of 60F	overcast, low of 50F, high of 60F
rain, low of 80F, high of 100F	rain, low of 40F, high of 60F	cloudy, low of 30F, high of 50F	snow, low of 20F, high of 40F
dew, low of 20F, high of 30F	cloudy, low of 90F, high of 100F	overcast, low of 50F, high of 70F	overcast, low of 80F, high of 90F
misty, low of 90F, high of 100F	hot, low of 70F, high of 90F	hail, low of 30F, high of 40F	rain, low of 60F, high of 80F
stormy, low of 90F, high of 100F	cloudy, low of 90F, high of 100F	clear skies, low of 90F, high of 100F	foggy, low of 30F, high of 50F
dry, low of 50F, high of 60F	foggy, low of 80F, high of 100F	overcast, low of 50F, high of 70F	overcast, low of 60F, high of 70F
rain, low of 30F, high of 50F	rain, low of 80F, high of 100F	dry, low of 50F, high of 60F	dry, low of 90F, high of 100F

DRIVER	Is it gonna snow in Durham during the week?
Seq2seq	there is no snow in the forecast for this week
+Att	there is no snow in the forecast for durham for durham
Ptr-Unk	yes there is no snow in durham on week
Mem2Seq	no snow, the forecast does not predict any snow on Durham
GOLD	Durham does not have any snow predicted this week

Table 5.9: Example of generated responses for the In-Car Assistant on the navigation domain.

distance	traffic_info	poi_type	address	poi
3 miles	no traffic	chinese restaurant	593 Arrowhead Way	Chef Chu's
1 miles	no traffic	chinese restaurant	669 El Camino Real	P.F. Changs
5 miles	road block nearby	gas station	200 Alester Ave	Valero
3 miles	no traffic	pizza restaurant	528 Anton Ct	Pizza My Heart
4 miles	heavy traffic	rest stop	753 University Ave	Comfort Inn
5 miles	heavy traffic	chinese restaurant	842 Arrowhead Way	Panda Express
2 miles	heavy traffic	pizza restaurant	704 El Camino Real	Pizza Hut
6 miles	no traffic	friends house	864 Almanor Ln	jacks house

DRIVER	Find me the nearest gas station.
Seq2seq	the nearest parking garage is from at away
+att	The is one is away away
PTRUNK	PAD nearest gas station is is is
Mem2Seq	The nearest gas station is gas station which is you you
GOLD	The nearest gas station is located 5 miles away. Need more info?

Table 5.10: Example of generated responses for the In-Car Assistant on the navigation domain.

Poi	Distance	Traffic_info	Poi_type	Address
The Westin	4 miles	no traffic	rest stop	329 El Camino Real
Palo Alto Garage R	2 miles	moderate traffic	parking garage	481 Amaranta Ave
Mandarin Roots	4 miles	moderate traffic	chinese restaurant	271 Springer Street
Sigona Farmers Market	4 miles	no traffic	grocery store	638 Amherst St
Whole Foods	2 miles	heavy traffic	grocery store	819 Alma St
Stanford Shopping Center	2 miles	moderate traffic	shopping center	773 Alger Dr
toms house	1 miles	no traffic	friends house	580 Van Ness Ave

DRIVER	Give me directions to the closest grocery store
CAR	There are Whole Foods 2 miles away and Sigona Farmers Market 4 miles away. Where do we go?
DRIVER	I need the one that avoids all heavy traffic
Seq2seq	he is is is is away at
+Att	Chef Chu s is located at 593 Arrowhead way
Ptr-Unk	PAD is the is PAD the to
Mem2Seq	Sigona Farmers Market is the only option i is 4 miles away with no traffic, setting navigation now
GOLD	Sigona Farmers Market is located 4 miles away with no traffic. It's located at 638 Amherst St.

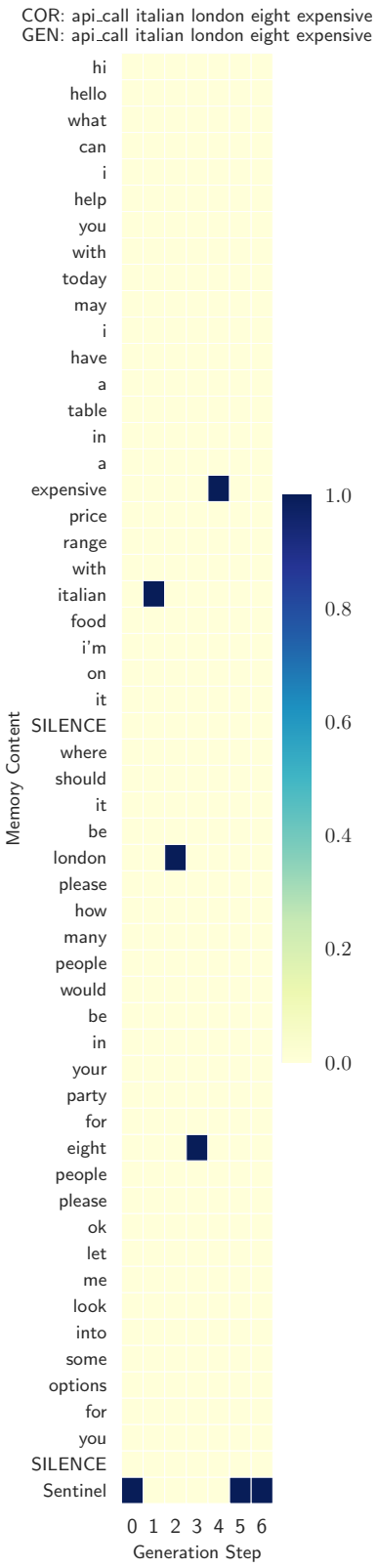


Figure 5.3: Last hop memory attention visualization from the bAbI dataset. COR and GEN on the top are the correct response and our generated one.

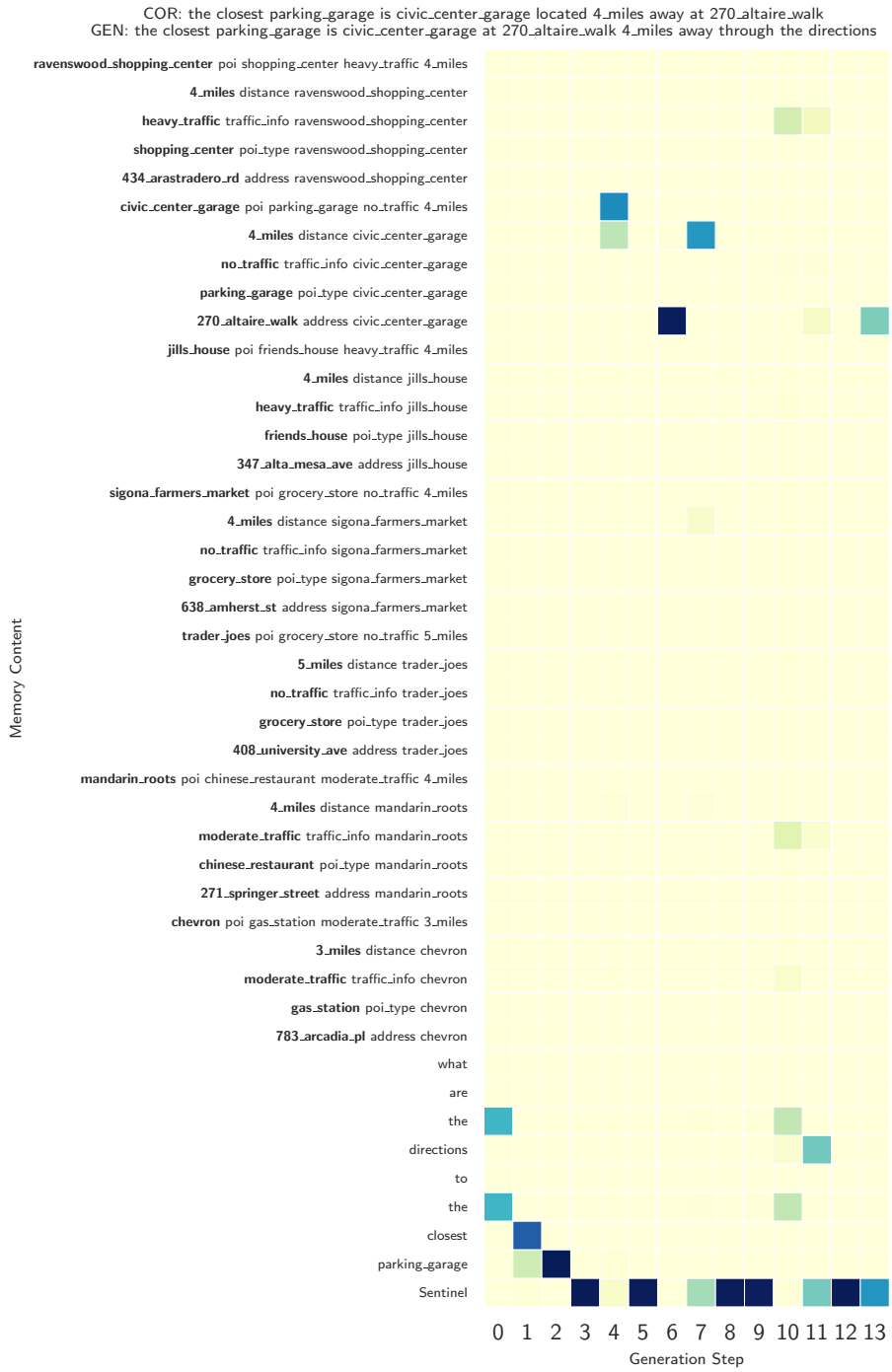


Figure 5.4: Mem2Seq memory attention visualization of last hop. Y-axis is the concatenation of KB information and dialogue history, and x-axis is the decoding step.

Visualization

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 5.4 and Figure 5.3 show the attention vector at the last hop for each generated token. Each column

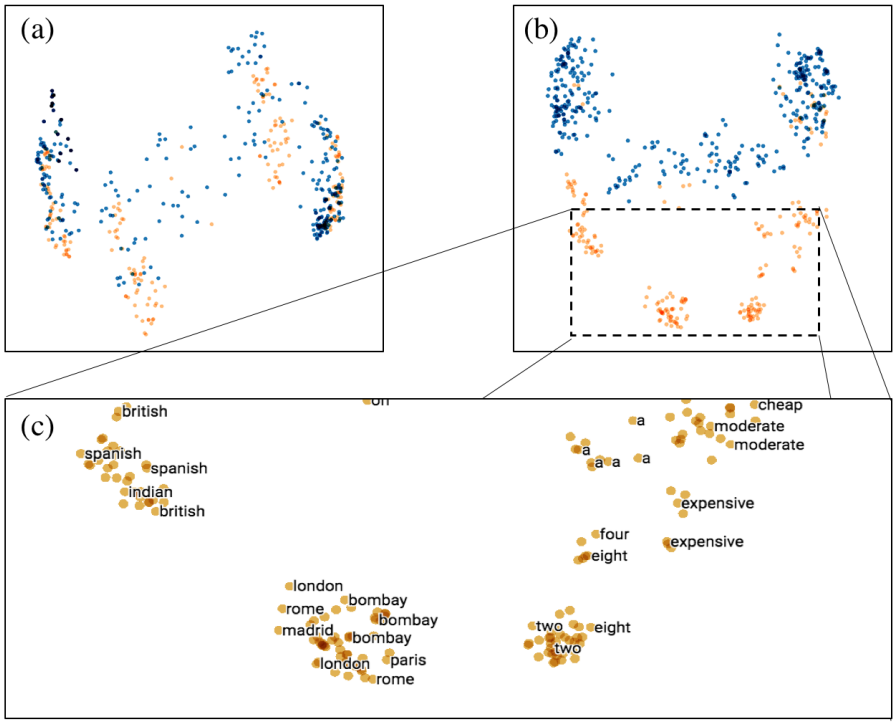


Figure 5.5: Principal component analysis of Mem2Seq query vectors in hop (a) 1 and (b) 6. (c) a closer look at clustered tokens.

represents the P_{ptr} vector at the corresponding generation step. Our model has a sharp distribution over the memory, which implies that it is able to select the right token from the memory. For example, the KB information "*270_altarie_walk*" was retrieved at the sixth step, which is an address for "*civic_center_garage*". On the other hand, if the sentinel is triggered, then the generated word comes from vocabulary distribution P_{vocab}. For instance, the third generation step triggers the sentinel, and "is" is generated from the vocabulary, as the word is not present in the dialogue history.

Query Vectors In Figure 5.5, the principal component analysis (PCA) of Mem2Seq query vectors are shown for different hops. Each dot is a query vector during each decoding time step, and it has a corresponding generated word. The blue dots are the words generated from P_{vocab}, which trigger the sentinel, and orange ones are from P_{ptr}. One can find that in (a), hop 1, there is no clear separation of the dots of the two different colors but they tend to group together. The separation becomes clearer in (b), hop 6, as dots of each color clusters into several groups, such as location, cuisine, and number. Our model tends to retrieve more information in the first hop, and points into the memories in the last hop.

Multiple Hops In Figure 5.6, Mem2Seq shows how multiple hops improve the model performance on several datasets. Task 3 in the bAbI dialogue dataset serves as an example, in which the systems need to recommend restaurants to users based on restaurant ranking from highest to lowest. Users can reject the recommendation and the system has to reason over the next highest restaurant. We find two common patterns between hops among different samples: 1) the first hop is usually used to score all the relevant memories and retrieve information; and 2) the last hop tends to focus on a specific token and makes mistakes when the attention is not sharp.

5.1.4 Short Summary

We present an end-to-end trainable memory-to-sequence (Mem2Seq) model for task-oriented dialogue systems. Mem2Seq combines the multi-hop attention mechanism in end-to-end memory networks with the idea of pointer networks to incorporate external information. It is a simple generative model that is able to incorporate KB information with promising generalization ability. We discover that the entity F1 score may be a more comprehensive evaluation metric than per-response accuracy or BLEU score, as humans can normally choose the right entities but have very diversified responses. Lastly, we empirically show our model's ability to produce relevant answers using both the external KB information and the predefined vocabulary, and visualize how the multi-hop attention mechanism helps in learning correlations between memories. Mem2Seq is fast, general, and able to achieve state-of-the-art results on three different datasets.

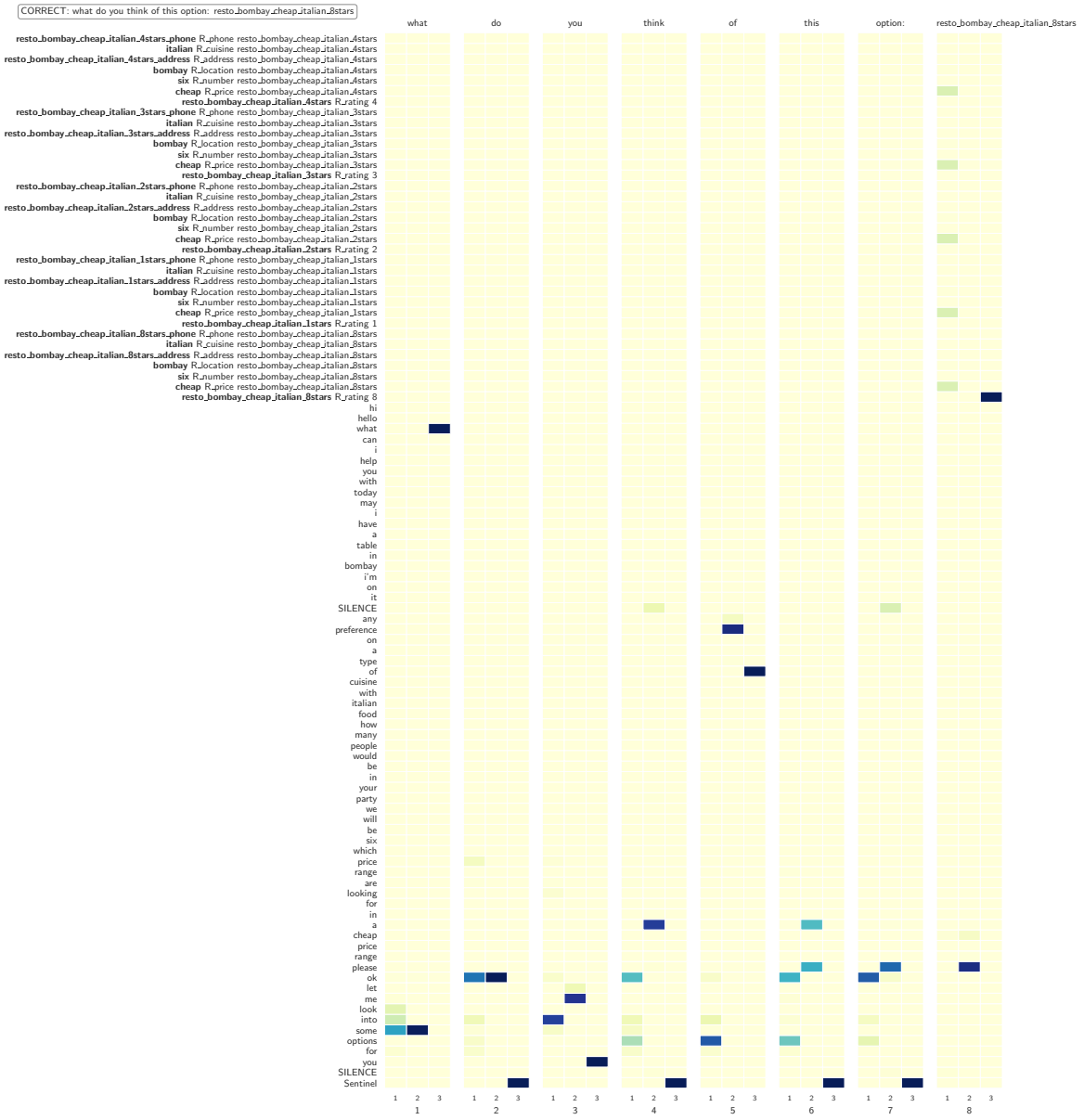


Figure 5.6: Mem2Seq multi-hop memory attention visualization. Each decoding step on the x-axis has three hops, from loose attention to sharp attention.

5.2 Global-to-Local Memory Pointer Networks

In the previous section, we discussed the first generative model that combines memory-augmented neural networks with copy mechanism to memorize long dialogue context and external knowledge. We empirically show the promising results of generated system responses in terms of BLEU score and entity F1 score on three different dialogue datasets. However, we found Mem2Seq tends to make the following errors while generating responses: 1) Wrong entity copying, as shown in Table 5.10. Although Mem2Seq can achieve the highest entity F1 score compared to existing baselines, it is only 33.4%. 2) The responses generated sometimes are not fluent, or even with several grammar mistakes, as shown in Table 5.9. Therefore, how to further improve the copy mechanism to obtain correct entity values becomes the next challenge, and how to maintain the fluency while balancing between generating from the vocabulary space and copying from the external knowledge.

In the section, we introduce the global-to-local memory pointer (GLMP) networks [85], which is an extension of Mem2Seq. GLMP sketches system responses with unfilled slots, strengthens the copy mechanism using double pointers, and sharing memory representation in external knowledge for encoder and decoder. This model is composed of a global memory encoder, a local memory decoder, and a shared external knowledge. Unlike existing approaches with copy ability [44, 53, 56, 84], in which the only information passed to the decoder is the encoder hidden states, GLMP shares the external knowledge and leverages the encoder and the external knowledge to learn a global memory pointer and global contextual representation. The global memory pointer modifies the external knowledge by softly filtering words that are not necessary for copying. Afterward, instead of generating system responses directly, the local memory decoder first uses a sketch RNN to obtain sketch responses without slot values but with sketch tags, which can be considered as learning latent dialogue management to generate dialogue action template. A similar intuition for generation sketching can be found in [92], [93] and [94]. Then the decoder generates local memory pointers to copy words from external knowledge and instantiate sketch tags.

We empirically show that GLMP can achieve superior performance using the combination of global and local memory pointers. In simulated OOV tasks on the bAbI dialogue dataset [39], GLMP achieves 92.0% per-response accuracy and surpasses existing end-to-end approaches by 7.5% on OOV full dialogue. On a human-human dialogue dataset [87], GLMP is able to surpass the previous state-of-the-art, including Mem2Seq, on both automatic and human evaluation.

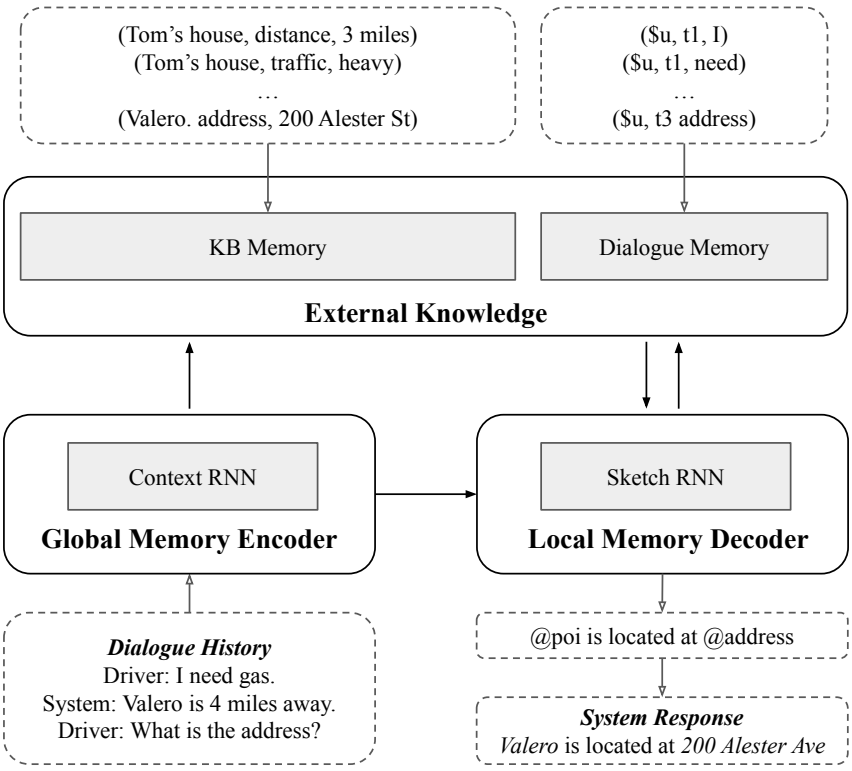


Figure 5.7: The block diagram of global-to-local memory pointer networks. There are three components: global memory encoder, shared external knowledge, and local memory decoder.

5.2.1 Model Description

The GLMP model [2] is composed of three parts: a global memory encoder, external knowledge, and local memory decoder, as shown in Figure 5.7. The dialogue history and the KB information are the input, and the system response is the expected output. First, the global memory encoder uses a context RNN to encode dialogue history and writes its hidden states into the external knowledge. Then the last hidden state is used to read the external knowledge and generate the global memory pointer at the same time. On the other hand, during the decoding stage, the local memory decoder first generates sketch responses by a sketch RNN. Then the global memory pointer and the sketch RNN hidden state are passed to the external knowledge as a filter and a query. The local memory pointer returned from the external knowledge can copy text from the external knowledge to replace the sketch tags and obtain the final system response.

[2]The code is available at `https://github.com/jasonwu0731/GLMP`

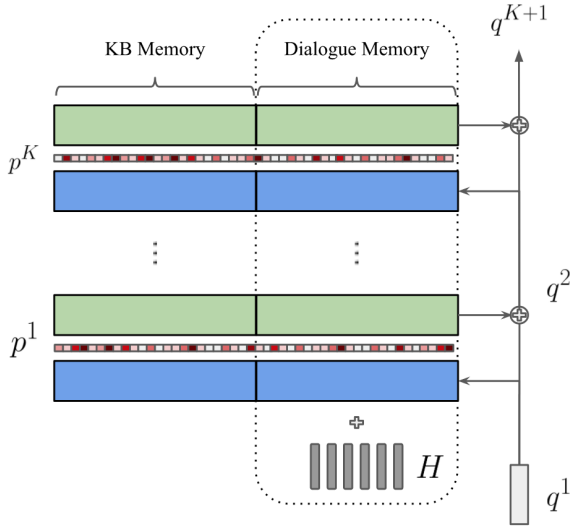


Figure 5.8: GLMP external knowledge architecture, KB memory and dialogue memory.

External Knowledge

The external knowledge contains the global contextual representation that is shared with the encoder and the decoder. To incorporate external knowledge into a learning framework, end-to-end memory networks are used to store word-level information for both structural KB (KB memory) and temporal-dependent dialogue history (dialogue memory), as shown in Figure 5.8. In addition, the MN is well-known for its multiple hop reasoning ability [12], which is appealing to strengthen the copy mechanism.

Global Contextual Representation In the KB memory module, each element is represented in the triplet format as a *(Subject, Relation, Object)* structure, which is a common format used to represent KB nodes [87, 95]. Meanwhile, the dialogue context is stored in the dialogue memory module, where the speaker and temporal encoding are included, as in [39], in a triplet format. For the two memory modules, a bag-of-words representation is used as the memory embeddings. As in the design in [84], during the inference time, we copy the object word once a memory position is pointed to. For example, *3 miles* will be copied if the triplet *(Toms house, distance, 3 miles)* is selected. We denote the $Object(.)$ function as getting the object word from a triplet.

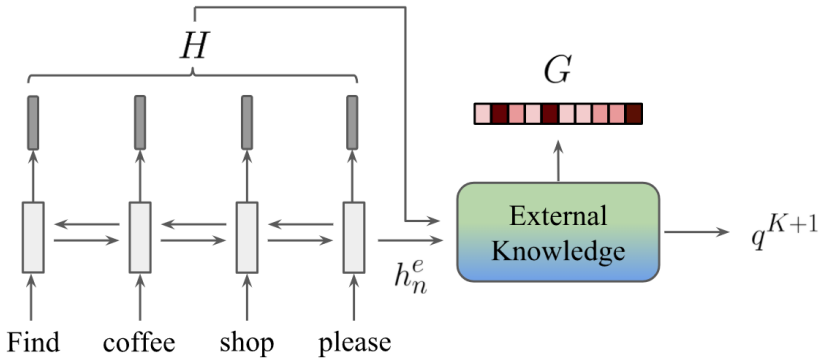


Figure 5.9: GLMP global memory encoder architecture.

Global Memory Encoder

In Figure 5.9, a context RNN is used to model the sequential dependency and encode the context. Then the hidden states are written into the external knowledge, as shown in Figure 5.8. Afterward, the last encoder hidden state serves as the query to read the external knowledge and get two outputs, the global memory pointer and the memory readout. Intuitively, since it is hard for MN architectures to model the dependencies between memories [79], which is a serious drawback, especially in conversational related tasks, writing the hidden states to the external knowledge can provide sequential and contextualized information. With meaningful representation, our pointers can correctly copy out words from the external knowledge, and the common OOV challenge can be mitigated. In addition, using the encoded dialogue context as a query can encourage our external knowledge to read out memory information related to the hidden dialogue states or user intention. Moreover, the global memory pointer that learns a global memory distribution is passed to the decoder along with the encoded dialogue history and KB information.

Context RNN A bi-directional GRU is used to encode dialogue history into the hidden states, and the last hidden state is used to query the external knowledge as the encoded dialogue history. In addition, the hidden states are written into the dialogue memory module in the external knowledge by summing up the original memory representation with the corresponding hidden states via:

$$c_i^k = c_i^k + h_{m_i}^{enc} \quad \text{if} \quad m_i \in X \text{ and } \forall k \in [1, K+1]. \tag{5.2}$$

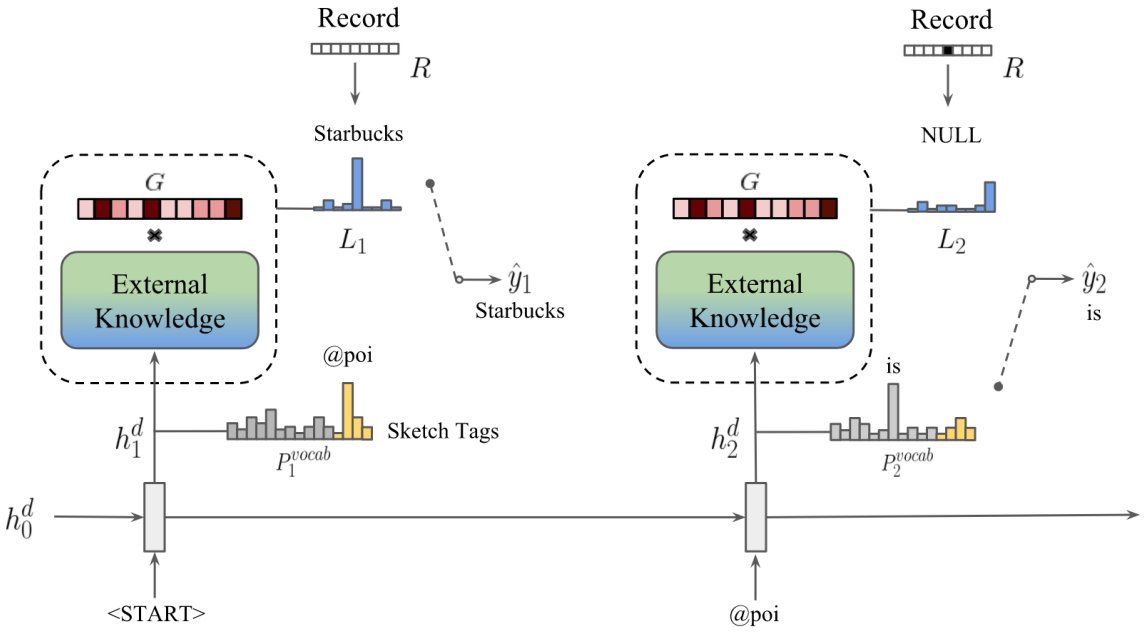


Figure 5.10: GLMP local memory decoder architecture.

Global memory pointer $G = (g_1, \ldots, g_M)$ is a vector containing real values between 0 and 1, where M is the total number of memories. Unlike the conventional attention mechanism in which all the weights sum to one, each element in G is an independent probability. We first query the external knowledge encoded dialogue history, and instead of applying the Softmax function as in (2.19), we perform an inner product followed by the Sigmoid function. The memory distribution we obtained is the global memory pointer G, which is passed to the decoder. To further strengthen the global pointing ability, we add an auxiliary loss to train the global memory pointer as a multi-label classification task. We show in an ablation study that adding this additional supervision does improve the performance. Lastly, the memory readout q^{K+1} is used as the encoded KB information.

In the auxiliary task, we define the label $G^{label} = (g_1^l, \ldots, g_M^l)$ by checking whether the object words in the memory exists in the expected system response Y. Then the global memory pointer is trained using the binary cross-entropy loss $Loss_g$ between G and G^{label}:

$$g_i = \text{Sigmoid}((q^K)^T c_i^K), \quad g_i^l = \begin{cases} 1 & \text{if } Object(m_i) \in Y \\ 0 & \text{otherwise} \end{cases}, \tag{5.3}$$

$$Loss_g = -\sum_{i=1}^{M} [g_i^l \times \log g_i + (1 - g_i^l) \times \log (1 - g_i)]. \tag{5.4}$$

Local Memory Decoder

Given the encoded dialogue history, the encoded KB information, and the global memory pointer, our local memory decoder first initializes its sketch RNN using the concatenation of the encoded dialogue history and KB information, and generates a sketch response that excludes slot values but includes the sketch tags. For example, sketch RNN will generate "*@poi is @distance away*" instead of "*Starbucks is 1 mile away.*" At each decoding time step, the hidden state of the sketch RNN is used for two purposes: 1) predicting the next token in the vocabulary, which is the same as standard sequence-to-sequence (Seq2Seq) learning; and 2) serving as the vector to query the external knowledge. If a sketch tag is generated, the global memory pointer is passed to the external knowledge, and the expected output word will be picked up from the local memory pointer. Otherwise, the output word is the word generated by the sketch RNN. For example in Figure 5.10, a @poi tag is generated at the first time step. Therefore, the word *Starbucks* is picked up from the local memory pointer as the system output word.

Sketch RNN We use a GRU to generate a sketch response $Y^s = (y_1^s, \ldots, y_{|Y|}^s)$ without real slot values. The sketch RNN learns to generate a dynamic dialogue action template based on the encoded dialogue and KB information. At each decoding time step t, the sketch RNN hidden state h_t^{dec} and its output distribution P_t^{vocab} are defined as

$$h_t^{dec} = \text{GRU}(C^1(\hat{y}_{t-1}^s), h_{t-1}^{dec}), \quad P_t^{vocab} = \text{Softmax}(W h_t^{dec}). \tag{5.5}$$

We use the standard cross-entropy loss to train the sketch RNN, and define $Loss_v$ as

$$Loss_v = \sum_{t=1}^{|Y|} -\log(P_t^{vocab}(y_t^s)). \tag{5.6}$$

We transform the slot values in Y into sketch tags based on the provided entity table. The sketch tags ST are all the possible slot types that start with a special token. For example, *@address* stands for all the addresses and *@distance* stands for all the distance information.

Local memory pointer $L = (L_1, \ldots, L_{|Y|})$ contains a sequence of pointers. At each time step t, the global memory pointer G first modifies the global contextual representation using its attention weights,

$$c_i^k = c_i^k \times g_i, \quad \forall i \in [1, M] \text{ and } \forall k \in [1, K+1], \tag{5.7}$$

and then the sketch RNN hidden state h_t^{dec} queries the external knowledge. The memory attention in the last hop is the corresponding local memory pointer L_t, which is represented as the memory distribution at time step t. To train the local memory pointer, supervision on top of the last hop memory attention in the external knowledge is added. We first define the position label of local memory pointer L^{label} at the decoding time step t as

$$L_t^{label} = \begin{cases} max(z) & \text{if } \exists z \text{ s.t. } y_t = Object(m_z), \\ M+1 & \text{otherwise.} \end{cases} \tag{5.8}$$

The position M+1 is a null token in the memory that allows us to calculate the loss function even if y_t does not exist in the external knowledge. Then, the loss between L and L^{label} is defined as

$$Loss_l = \sum_{t=1}^{|Y|} -\log(L_t(L_t^{label})). \tag{5.9}$$

Furthermore, a record $R \in \mathbb{R}^{n+l}$ is utilized to prevent copying the same entities multiple times. All the elements in R are initialized as 1 in the beginning. During the decoding stage, if a memory position has been pointed to, its corresponding position in R will be masked out. During the inference time, $\hat{y}_t$ is defined as

$$\hat{y}_t = \begin{cases} \arg\max(P_t^{vocab}) & \text{if } \arg\max(P_t^{vocab}) \notin ST, \\ Object(m_{\arg\max(L_t \odot R)}) & \text{otherwise,} \end{cases} \tag{5.10}$$

where $\odot$ is the element-wise multiplication. Lastly, all the parameters are jointly trained by minimizing the weighted-sum of three losses (α, β, γ are hyper-parameters):

$$Loss = \alpha Loss_g + \beta Loss_v + \gamma Loss_l. \tag{5.11}$$

5.2.2 Experimental Setup

Training

The model is trained end-to-end using the Adam optimizer [69], and learning rate annealing starts from $1e^{-3}$ to $1e^{-4}$. The number of hops K is set to 1, 3, or 6 to compare the performance difference. The weights α, β, and γ summing the three losses, are set to 1. All the embeddings are initialized randomly, and a simple greedy strategy is used without beam-search during the decoding stage. The hyper-parameters such as hidden size and dropout rate are tuned with grid-search over the development set (per-response accuracy for bAbI dialogue and BLEU score

for the In-Car Assistant). In addition, to increase model generalization and simulate an OOV setting, we randomly mask a small number of input source tokens into an unknown token.

5.2.3 Results and Discussion

bAbI Dialogue

Table 5.11: GLMP per-response accuracy and completion rate on bAbI dialogues.

Task	MN	GMN	S2S+Attn	Ptr-Unk	Mem2Seq	GLMP H1	GLMP H3	GLMP H6
T1	99.9 (99.6)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T2	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T3	74.9 (2.0)	74.9 (0)	74.8 (0)	85.1 (19.0)	94.7 (62.1)	**96.3 (75.6)**	96.0 (69.4)	96.0 (68.7)
T4	59.5 (3.0)	57.2 (0)	57.2 (0)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T5	96.1 (49.4)	96.3 (52.5)	98.4 (87.3)	99.4 (91.5)	97.9 (69.6)	99.2 (88.5)	99.0 (86.5)	99.2 (89.7)
T1 oov	72.3 (0)	82.4 (0)	81.7 (0)	92.5 (54.7)	94.0 (62.2)	**100 (100)**	**100 (100)**	99.3 (95.9)
T2 oov	78.9 (0)	78.9 (0)	78.9 (0)	83.2 (0)	86.5 (12.4)	**100 (100)**	**100 (100)**	99.4 (94.6)
T3 oov	74.4 (0)	75.3 (0)	75.3 (0)	82.9 (13.4)	90.3 (38.7)	95.5 (65.7)	**96.7 (72.9)**	95.9 (67.7)
T4 oov	57.6 (0)	57.0 (0)	57.0 (0)	100 (100)	100 (100)	100 (100)	100 (100)	100 (100)
T5 oov	65.5 (0)	66.7 (0)	65.7 (0)	73.6 (0)	84.5 (2.3)	**92.0 (21.7)**	91.0 (17.7)	91.8 (21.4)

In Table 5.11, we follow Bordes et al. [39] to compare the performance of our model and baselines based on per-response accuracy and task-completion rate. Note that utterance retrieval methods, such as QRN, MN, and GMN, cannot correctly recommend options (T3) and provide additional information (T4), and a poor generalization ability is observed in the OOV setting, which shows around 30% performance difference on Task 5. Although previous generation-based approaches (Ptr-Unk, Mem2Seq) mitigate the gap by incorporating a copy mechanism, the simplest cases such as generating and modifying API calls (T1, T2) still face a 6–17% OOV performance drop. On the other hand, GLMP achieves the highest, 92.0%, task-completion rate on the full dialogue task and surpasses baselines by a big margin, especially in the OOV setting. No per-response accuracy loss cab be seen for T1, T2, and T4 using only the single hop, and it only decreases 7–9% on the full dialogue task.

In-Car Assistant

For a human-human dialogue scenario, we follow previous dialogue works [80, 84, 87] to evaluate our system on two automatic evaluation metrics, BLEU and entity F1 score. As shown in Table 5.12, GLMP achieves the highest BLEU and entity F1 scores of 14.79 and 59.97% respectively, which represent a slight improvement in BLEU but a huge gain in entity F1. In

Table 5.12: GLMP performance on In-Car Assistant dataset using automatic evaluation (BLEU and entity F1) and human evaluation (appropriate and humanlike).

Automatic Evaluation

	Rule-Based*	KVR*	S2S	S2S + Attn	Ptr-Unk	Mem2Seq	GLMP K1	GLMP K3	GLMP K6
BLEU	6.6	13.2	8.4	9.3	8.3	12.6	13.83	**14.79**	12.37
Entity F1	43.8	48.0	10.3	19.9	22.7	33.4	57.25	**59.97**	53.54
Schedule F1	61.3	62.9	9.7	23.4	26.9	49.3	68.74	**69.56**	69.38
Weather F1	39.5	47.0	14.1	25.6	26.7	32.8	60.87	**62.58**	55.89
Navigation F1	40.4	41.3	7.0	10.8	14.9	20.0	48.62	**52.98**	43.08

Human Evaluation

	Mem2Seq	GLMP	Human
Appropriate	3.89	4.15	4.6
Humanlike	3.80	4.02	4.54

fact, for unsupervised evaluation metrics in task-oriented dialogues, we argue that the entity F1 might be a more comprehensive evaluation metric than per-response accuracy or BLEU, as it is shown in [87] that humans are able to choose the right entities but have very diversified responses. Note that the results of rule-based and KVR are not directly comparable because they simplify the task by mapping the expression of entities to a canonical form using named entity recognition and linking. For example, they compared in "*@poi is @poi_distance away*," instead of "*Starbucks is 1_mile away.*"

Moreover, human evaluation of the generated responses is reported. We compare GLMP with the state-of-the-art model Mem2Seq and the original dataset responses as well. We randomly select 200 different dialogue scenarios from the test set to evaluate three different responses. Amazon Mechanical Turk is used to evaluate system appropriateness and humanlikeness on a scale from 1 to 5. With the results shown in Table 5.12, we can see that GLMP outperforms Mem2Seq on both measures, which is consistent with the previous observation. We also see that human performance on this assessment sets the upper bound on scores, as expected.

Ablation Study

The contributions of the global memory pointer G and the memory writing of dialogue history H are shown in Table 5.13. We compare the results using GLMP with one hop in the bAbI OOV setting and In-Car Assistant. GLMP without H means that the context RNN in the global memory encoder does not write the hidden states into external knowledge. As one can observe, GLMP without H has 5.3% more loss on the full dialogue task. On the other hand, GLMP

Table 5.13: Ablation study using single hop model. Numbers in parentheses indicate how seriously the performance drops.

	bAbI Dialogue OOV Per-response Accuracy					In-Car Assistant Entity F1
	T1	T2	T3	T4	T5	All
GLMP	100 (-)	100 (-)	95.5 (-)	100 (-)	92.0 (-)	57.25 (-)
GLMP w/o H	90.4 (-9.6)	85.6 (-14.4)	95.4 (-0.1)	100 (-0)	86.2 (-5.3)	47.96 (-9.29)
GLMP w/o G	100 (-0)	91.7 (-8.3)	95.5 (-0)	100 (-0)	92.4 (+0.4)	45.78 (-11.47)

without G means that we do not use the global memory pointer to modify the external knowledge, and an 11.47% entity F1 drop can be observed on the In-Car Assistant dataset. Note that a 0.4% increase can be observed on task 5, which suggests that the use of the global memory pointer may impose too strong prior entity probability.

Visualization and Qualitative Evaluation

Analyzing the attention weights has been frequently used to interpret deep learning models. In Figure 5.11, Figure 5.12, and Figure 5.13, we show the attention vector in the last hop for each generation time step. The Y-axis is the external knowledge that we can copy, including the KB information and dialogue history. In Figure 5.11, based on the question "*What is the address?*" asked by the driver in the last turn, the gold answer and our generated response are on the top, and the global memory pointer G is shown in the left column. One can observe that in the right column, the final memory pointer has successfully copied the entity *chevron* in step 0 and its address *783 Arcadia Pl* in step 3 to fill in the sketch utterance. Memory attention without global weighting is reported in the middle column, and one can find that even if the attention weights focus on several points of interest and addresses in step 0 and step 3, the global memory pointer can mitigate the issue as expected.

Error Analysis

We also check the main mistakes made by our model and analyze the errors. For the bAbI dialogues, the mistakes are mainly from task 3, which is recommending restaurants based on their rating from high to low. We find that sometimes the system will keep sending restaurants with a higher score even if the user has rejected them in the previous turns. Meanwhile, the In-Car Assistant is more challenging for response generation. First, we find that the model makes mistakes when the KB has several options corresponding to the user intention. For example, once the user has more than one doctor's appointment in the table, the model can barely

recognize them. In addition, since we do not include the domain specific and user intention supervision, wrong delexicalized responses may be generated, which results in an incorrect entity copy. Lastly, we find that the copied entities may not match the generated sketch tags. For example, an address tag may result in a totally unrelated entity copy such as "4 miles".

5.2.4 Short Summary

We present an end-to-end trainable model called a global-to-local memory pointer network (GLMP) for task-oriented dialogues to effectively incorporate long dialogue context and external knowledge. It is an extension of the Mem2Seq generation model, to address the wrong entity copying and response fluency problems. The global memory encoder and the local memory decoder are designed to incorporate shared external knowledge into the learning framework. We empirically show that the global and the local memory pointer are able to effectively produce system responses even in the out-of-vocabulary scenario, and visualize how the global memory pointer helps as well. As a result, our model achieves state-of-the-art results on both simulated and human-human dialogue datasets.

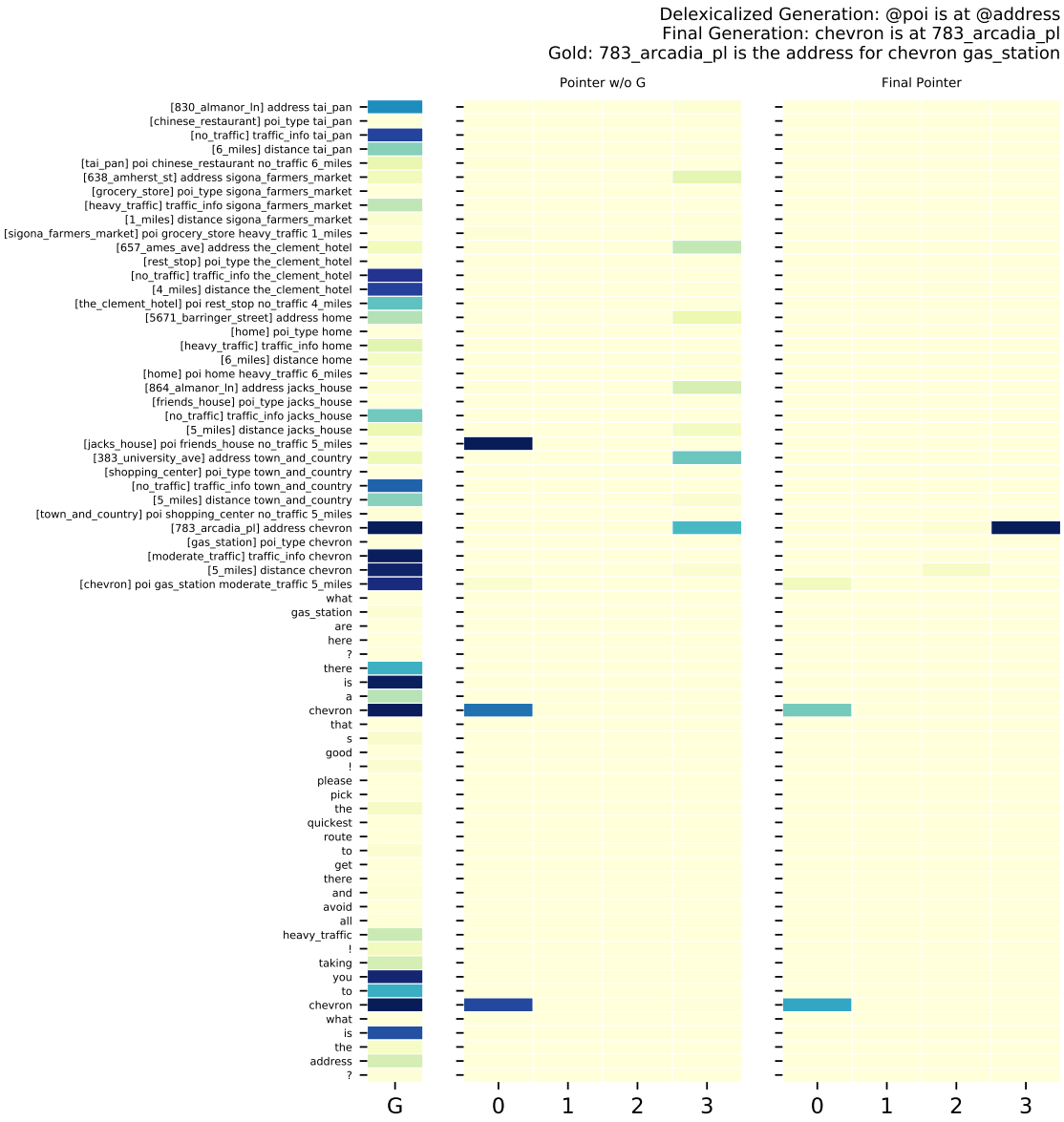


Figure 5.11: Memory attention visualization in the In-Car Assistant dataset on navigation domain. The left column is the memory attention of global memory pointer, the right column is the local memory pointer over four decoding steps. The middle column is the local memory pointer without weighted by global memory pointer.

Figure 5.12: Memory attention visualization in the In-Car Assistant dataset on schedule domain.

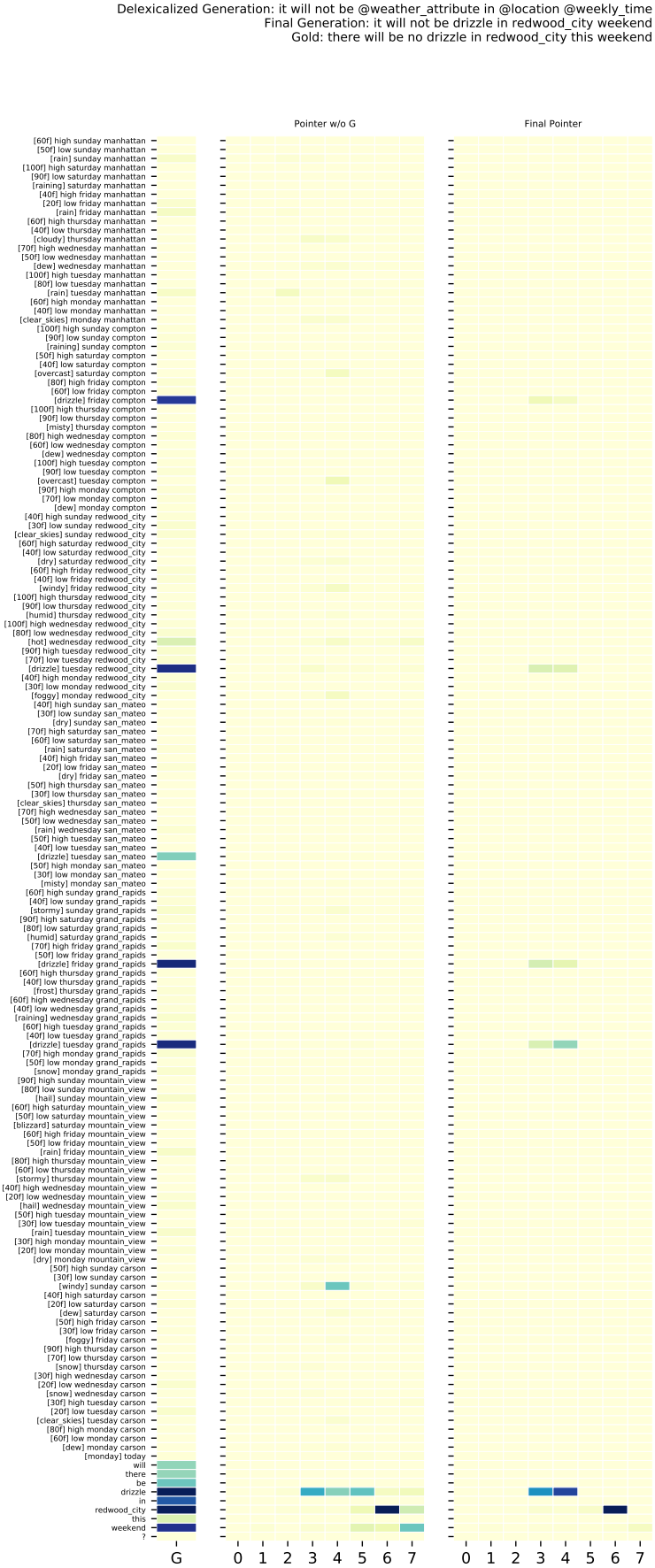


Figure 5.13: Memory attention visualization in the In-Car Assistant dataset on weather domain.

Chapter 6

Conclusion

In this thesis, we focus on learning task-oriented dialogue systems with deep learning models. We pointed out challenges in existing approaches to modeling long dialogue context and external knowledge in conversation and optimizing dialogue systems end-to-end. To effectively address these challenges, we incorporated and strengthened the neural copy mechanism with memory-augmented neural networks, and leveraged this strategy to achieve state-of-the-art performance in multi-domain dialogue state tracking, retrieval-based dialogue systems, and generation-based dialogue systems. In this chapter, we conclude the thesis and discuss possible future work.

In Chapter 3, we showed an end-to-end generative model with a copy mechanism can be used in dialogue state tracking, and sharing in multiple domains can further improve the performance. Our proposed dialogue state generator (TRADE) achieved state-of-the-art results in multi-domain dialogue state tracking. We also demonstrated how to track unseen domains by transferring knowledge from learned domains, or to quickly adapt to a new domain without forgetting the learned domains.

In Chapter 4, we leveraged recurrent entity networks and proposed dynamic query memory networks for end-to-end retrieval-based dialogue learning. We obtained state-of-the-art per-response/per-dialogue accuracy via modeling sequential dependencies and external knowledge using memory-augmented neural networks. In addition, we demonstrated how a simple copy mechanism, recorded delexicalized copying, can be used to reduce learning complexity and improve model generalization ability.

In Chapter 5, we introduced two neural models with a copy mechanism that achieved state-of-the-art performance on end-to-end response generation tasks. We presented the first machine learning model (Mem2Seq) that combines a multi-hop attention mechanism with the idea of pointer networks. Then we presented its extension model (GLMP), which strengthens the

copying accuracy with double pointers and response sketching. We showed that both models outperform existing generation approaches, not only in terms of automatic evaluation metrics, such as BLEU and F1 score, but also on human evaluation metrics, such as appropriateness and human likeness.

Finally, conventional task-oriented dialogue systems [3], which are still widely used in commercial systems, require significant amounts human effort in system design and data collection. End-to-end dialogue systems, although not perfect yet, require much less human involvement, especially in the dataset construction, as raw conversational text and KB information can be used directly without the need of heavy pre-processing, e.g., named-entity recognition and dependency parsing.

In future works, several methods could be applied (e.g. Reinforcement Learning [96], Beam Search [97]) to improve both responses relevance and entity F1 score. However, we preferred to keep our model as simple as possible in order to show that it works well even without advanced training methods. Also it is possible to further improve the copying accuracy using different kinds of memory-augmented neural networks or using hierarchical structures to store external knowledge. In multi-domain dialogue state tracking, we expect to further improve zero-shot or few-shot learning by collecting a larger dataset with more domains. We believe end-to-end systems can be further improved by taking steps toward multi-task training as well, e.g., joint training dialogue state tracking in response generation tasks. It will be interesting to see how the concept of conventional pipeline dialogue approaches can help the design of end-to-end dialogue learning in task-oriented dialogue systems.

List of Publications

(* denotes equal contribution)

1. Chien-Sheng Wu, Andrea Madotto, Zhaojiang Lin, Peng Xu, and Pascale Fung. "Getting To Know You: Extracting User Attributes from Conversations for Personalized Dialogue Agents." (Under Review)

2. Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. "Transferable Multi-Domain Dialogue State Generators for Task-Oriented Dialogue Systems." In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*

3. Chien-Sheng Wu, Richard Socher, and Caiming Xiong. "Global-to-local Memory Pointer Networks for Task-Oriented Dialogue" In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

4. Chien-Sheng Wu*, Andrea Madotto*, and Pascale Fung. "Mem2Seq: Effectively Incorporating Knowledge Bases into End-to-End Task-Oriented Dialog Systems." In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)* (Volume 1: Long Papers). Vol. 1. 2018.

5. Chien-Sheng Wu, Andrea Madotto, Genta Winata, and Pascale Fung. "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)*, pp. 6154-6158. IEEE, 2018.

6. Chien-Sheng Wu*, Andrea Madotto*, Genta Winata, and Pascale Fung. "End-to-end recurrent entity network for entity-value independent goal-oriented dialog learning." In *Dialog System Technology Challenges Workshop, DSTC6*. 2017.

References

[1] C. Raymond and G. Riccardi, "Generative and discriminative algorithms for spoken language understanding," in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[2] L. Deng, G. Tur, X. He, and D. Hakkani-Tur, "Use of kernel deep convex networks and end-to-end learning for spoken language understanding," in *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2012, pp. 210–215.

[3] J. D. Williams and S. Young, "Partially observable markov decision processes for spoken dialog systems," *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.

[4] B. Thomson and S. Young, "Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems," *Computer Speech & Language*, vol. 24, no. 4, pp. 562–588, 2010.

[5] M. Henderson, B. Thomson, and S. Young, "Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 360–365.

[6] A. Rudnicky and W. Xu, "An agenda-based dialog management architecture for spoken language systems," in *IEEE Automatic Speech Recognition and Understanding Workshop*, vol. 13, no. 4, 1999.

[7] S. Young, "Using pomdps for dialog management," in *2006 IEEE Spoken Language Technology Workshop*. IEEE, 2006, pp. 8–13.

[8] S. Young, M. Gašić, B. Thomson, and J. D. Williams, "Pomdp-based statistical spoken dialog systems: A review," *Proceedings of the IEEE*, vol. 101, no. 5, pp. 1160–1179, 2013.

[9] S. Busemann and H. Horacek, "A flexible shallow approach to text generation," *arXiv preprint cs/9812018*, 1998.

[10] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[11] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[12] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[13] K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 189–194.

[14] D. Guo, G. Tur, W.-t. Yih, and G. Zweig, "Joint semantic utterance classification and slot filling with recursive neural networks," in *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2014, pp. 554–559.

[15] X. Zhang and H. Wang, "A joint model of intent determination and slot filling for spoken language understanding." in *IJCAI*, 2016, pp. 2993–2999.

[16] G. Tur, L. Deng, D. Hakkani-Tür, and X. He, "Towards deeper understanding: Deep convex networks for semantic utterance classification," in *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012, pp. 5045–5048.

[17] Y.-N. Chen, D. Hakkani-Tür, and X. He, "Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6045–6049.

[18] N. Nguyen and Y. Guo, "Comparisons of sequence labeling algorithms and extensions," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 681–688.

[19] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu *et al.*, "Using recurrent neural networks for slot filling in spoken language

understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 530–539, 2015.

[20] G. Kurata, B. Xiang, B. Zhou, and M. Yu, "Leveraging sentence-level information with encoder lstm for semantic slot filling," *arXiv preprint arXiv:1601.01530*, 2016.

[21] J. Williams, A. Raux, and M. Henderson, "The dialog state tracking challenge series: A review," *Dialogue & Discourse*, vol. 7, no. 3, pp. 4–33, 2016.

[22] N. Mrkšić, D. Ó Séaghdha, T.-H. Wen, B. Thomson, and S. Young, "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)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1777–1788.

[23] A. Rastogi, D. Hakkani-Tür, and L. Heck, "Scalable multi-domain dialogue state tracking," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 561–568.

[24] P. Xu and Q. Hu, "An end-to-end approach for handling unknown slot values in dialogue state tracking," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 1448–1457.

[25] O. Ramadan, P. Budzianowski, and M. Gasic, "Large-scale multi-domain belief tracking with knowledge sharing," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, 2018, pp. 432–437.

[26] V. Zhong, C. Xiong, and R. Socher, "Global-locally self-attentive dialogue state tracker," in *Association for Computational Linguistics*, 2018.

[27] L. Li, J. D. Williams, and S. Balakrishnan, "Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection," in *Tenth Annual Conference of the International Speech Communication Association*, 2009.

[28] Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and L. Deng, "Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] J. Gao, M. Galley, and L. Li, "Neural approaches to conversational ai," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 2018, pp. 1371–1374.

[30] T.-H. Wen, M. Gasic, N. Mrksic, P.-H. Su, D. Vandyke, and S. Young, "Semantically conditioned lstm-based natural language generation for spoken dialogue systems," *arXiv preprint arXiv:1508.01745*, 2015.

[31] O. Press, A. Bar, B. Bogin, J. Berant, and L. Wolf, "Language generation with recurrent generative adversarial networks without pre-training," *arXiv preprint arXiv:1706.01399*, 2017.

[32] L. Shang, Z. Lu, and H. Li, "Neural responding machine for short-text conversation," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, July 2015, pp. 1577–1586.

[33] P. Fung, A. Dey, F. B. Siddique, R. Lin, Y. Yang, D. Bertero, Y. Wan, R. H. Y. Chan, and C.-S. Wu, "Zara: A virtual interactive dialogue system incorporating emotion, sentiment and personality recognition," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, 2016, pp. 278–281.

[34] P. Fung, D. Bertero, Y. Wan, A. Dey, R. H. Y. Chan, F. B. Siddique, Y. Yang, C.-S. Wu, and R. Lin, "Towards empathetic human-robot interactions," *arXiv preprint arXiv:1605.04072*, 2016.

[35] P. Fung, D. Bertero, P. Xu, J. H. Park, C.-S. Wu, and A. Madotto, "Empathetic dialog systems," in *Proceedings of the International Conference on Language Resources and Evaluation*. European Language Resources Association, 2018.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[38] T.-H. Wen, D. Vandyke, N. Mrksic, M. Gasic, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young, "A network-based end-to-end trainable task-oriented dialogue system," *European Association of Computational Linguistics*, 2016.

[39] A. Bordes, Y.-L. Boureau, and J. Weston, "Learning end-to-end goal-oriented dialog," *International Conference on Learning Representations*, 2016.

[40] J. Perez and F. Liu, "Gated end-to-end memory networks," in *European Chapter of the Association for Computational Linguistics*, 2017.

[41] M. Seo, S. Min, A. Farhadi, and H. Hajishirzi, "Query-reduction networks for question answering," *International Conference on Learning Representations*, 2017.

[42] J. D. Williams, K. Asadi, and G. Zweig, "Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning," in *Association for Computational Linguistics*, 2017.

[43] W. Lei, X. Jin, M.-Y. Kan, Z. Ren, X. He, and D. Yin, "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)*, vol. 1, 2018, pp. 1437–1447.

[44] M. Eric and C. Manning, "A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Valencia, Spain: Association for Computational Linguistics, April 2017, pp. 468–473.

[45] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[46] K. Cho, B. van Merrienboer, aglar Gülehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[47] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. C. Courville, and Y. Bengio, "A hierarchical latent variable encoder-decoder model for generating dialogues." in *AAAI*, 2017, pp. 3295–3301.

[48] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[49] P. Xu, A. Madotto, C.-S. Wu, J. H. Park, and P. Fung, "Emo2vec: Learning generalized emotion representation by multi-task training," in *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2018, pp. 292–298.

[50] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[51] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Conference on Empirical Methods in Natural Language Processing*, 2015.

[52] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Neural Information Processing Systems*, 2015.

[53] C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio, "Pointing the unknown words," *arXiv preprint arXiv:1603.08148*, 2016.

[54] M. Dehghani, S. Rothe, E. Alfonseca, and P. Fleury, "Learning to attend, copy, and generate for session-based query suggestion," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. ACM, 2017.

[55] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *International Conference on Learning Representations*, 2016.

[56] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1631–1640.

[57] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," *CoRR*, vol. abs/1806.08730, 2018.

[58] S. He, C. Liu, K. Liu, and J. Zhao, "Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 199–208.

[59] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *International Conference on Machine Learning*, 2016, pp. 1378–1387.

[60] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, p. 471, 2016.

[61] M. Wang, Z. Lu, H. Li, and Q. Liu, "Memory-enhanced decoder for neural machine translation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 278–286.

[62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[63] L. Kaiser, O. Nachum, A. Roy, and S. Bengio, "Learning to remember rare events," *International Conference on Learning Representations*, 2017.

[64] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv preprint arXiv:1502.05698*, 2015.

[65] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, "Transferable multi-domain state generator for task-oriented dialogue systems," in *Proceedings of the 57rd Annual Meeting of the Association for Computational Linguistics (Long Papers)*. Association for Computational Linguistics, 2019.

[66] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, M. Hughes, and J. Dean, "Google's multilingual neural machine translation system: Enabling zero-shot translation," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 339–351, 2017.

[67] M. Henderson, B. Thomson, and J. D. Williams, "The second dialog state tracking challenge," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014, pp. 263–272.

[68] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic, "Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 5016–5026.

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

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

[71] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A joint many-task model: Growing a neural network for multiple nlp tasks," *arXiv preprint arXiv:1611.01587*, 2016.

[72] E. Nouri and E. Hosseini-Asl, "Toward scalable neural dialogue state tracking model," in *Advances in neural information processing systems (NeurIPS), 2nd Conversational AI workshop*, 2018.

[73] A. Bapna, G. Tur, D. Hakkani-Tur, and L. Heck, "Towards zero-shot frame semantic parsing for domain scaling," *arXiv preprint arXiv:1707.02363*, 2017.

[74] M. Gašić and S. Young, "Gaussian processes for pomdp-based dialogue manager optimization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 1, pp. 28–40, 2014.

[75] T. Zhao and M. Eskenazi, "Zero-shot dialog generation with cross-domain latent actions," in *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, 2018, pp. 1–10.

[76] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, p. 201611835, 2017.

[77] D. Lopez-Paz *et al.*, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[78] M. Henaff, J. Weston, A. Szlam, A. Bordes, and Y. LeCun, "Tracking the world state with recurrent entity networks," *International Conference on Learning Representations*, vol. abs/1612.03969, 2017.

[79] C.-S. Wu, A. Madotto, G. Winata, and P. Fung, "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)*, April 2018, pp. 6154–6158.

[80] T. Zhao, A. Lu, K. Lee, and M. Eskenazi, "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*. Association for Computational Linguistics, August 2017, pp. 27–36.

[81] A. Madotto and G. Attardi, "Question dependent recurrent entity network for question answering," *NL4AI*, 2017.

[82] C.-S. Wu, A. Madotto, G. Winata, and P. Fung, "End-to-end recurrent entity network for entity-value independent goal-oriented dialog learning," in *Dialog System Technology Challenges Workshop*, 2017.

[83] J. Perez, Y.-L. Boureau, and A. Bordes, "Dialog system & technology challenge 6 overview of track 1-end-to-end goal-oriented dialog learning."

[84] A. Madotto, C.-S. Wu, and P. Fung, "Mem2seq: Effectively incorporating knowledge bases into end-to-end task-oriented dialog systems," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 1468–1478.

[85] C.-S. Wu, R. Socher, and C. Xiong, "Global-to-local memory pointer networks for task-oriented dialogue," *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[86] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 1400–1409.

[87] M. Eric, L. Krishnan, F. Charette, and C. D. Manning, "Key-value retrieval networks for task-oriented dialogue," in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, 2017, pp. 37–49.

[88] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1073–1083.

[89] M. Henderson, B. Thomson, and J. D. Williams, "The second dialog state tracking challenge," in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2014, pp. 263–272.

[90] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. [Online]. Available: http://www.aclweb.org/anthology/P02-1040

[91] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 110–119.

[92] S. Mori, H. Maeta, T. Sasada, K. Yoshino, A. Hashimoto, T. Funatomi, and Y. Yamakata,

"Flowgraph2text: Automatic sentence skeleton compilation for procedural text generation," in *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, 2014, pp. 118–122.

[93] L. Dong and M. Lapata, "Coarse-to-fine decoding for neural semantic parsing," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018, pp. 731–742. [Online]. Available: http://aclweb.org/anthology/P18-1068

[94] D. Cai, Y. Wang, V. Bi, Z. Tu, X. Liu, W. Lam, and S. Shi, "Skeleton-to-response: Dialogue generation guided by retrieval memory," *arXiv preprint arXiv:1809.05296*, 2018.

[95] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, November 2016, pp. 1400–1409. [Online]. Available: https://aclweb.org/anthology/D16-1147

[96] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.

[97] S. Wiseman and A. M. Rush, "Sequence-to-sequence learning as beam-search optimization," *arXiv preprint arXiv:1606.02960*, 2016.