

Sahisnu Mazumder · Bing Liu

Lifelong and Continual Learning Dialogue Systems

Synthesis Lectures on Human Language Technologies

Series Editor

Graeme Hirst, Department of Computer Science, University of Toronto, Toronto, ON,
Canada

The series publishes topics relating to natural language processing, computational linguistics, information retrieval, and spoken language understanding. Emphasis is on important new techniques, on new applications, and on topics that combine two or more HLT subfields.

Sahisnu Mazumder · Bing Liu

Lifelong and Continual Learning Dialogue Systems

 Springer

Sahisnu Mazumder
Intel Labs
Santa Clara, CA, USA

Bing Liu
University of Illinois Chicago
Chicago, IL, USA

ISSN 1947-4040 ISSN 1947-4059 (electronic)
Synthesis Lectures on Human Language Technologies
ISBN 978-3-031-48188-8 ISBN 978-3-031-48189-5 (eBook)
<https://doi.org/10.1007/978-3-031-48189-5>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

Sahisnu dedicates this book to his parents Ramesh Ch. Mazumder and Dulali Mazumder, wife Mohana, sister Snigdha, brother-in-law Sudip, and (late) aunt Malaya Goswami.

Bing dedicates this book to his wife Yue He, his children, Shelley and Kate, and his parents Yuhan Liu and Xianzhen Dong.

Preface

The purpose of writing this book is to introduce the emerging topic of lifelong/continual learning dialogue systems. Dialogue systems, also commonly known as chatbots, are computer programs that can converse with humans to perform intended tasks. They typically employ text or speech modes. As deep learning has improved the accuracy of speech recognition, text generation, and processing dramatically since 2012 or so, dialogue systems are becoming increasingly ubiquitous and are used in all types of applications such as smartphones, cars, home appliances, company websites, and mobile robots. They provide a very rich set of services, e.g., performing some specific tasks and chit-chatting with human users. However, the user experiences have not been very satisfactory so far (discussed in Chap. 1). It is clearly unfair to expect a deployed dialogue system to be able to understand everything that users may say, but these systems should be able to learn during conversations by interacting with users to become more and more knowledgeable and powerful. That is the goal of building lifelong learning dialogue systems. That is also the key motivation for us to write this book to introduce and promote the research of lifelong learning dialogue systems.

The project of writing this book started with a tutorial titled *Continual Learning Dialogue Systems—Learning on the Job after Model Deployment* that we gave at the *2021 International Joint Conference on Artificial Intelligence (IJCAI-2021)*, August 21–26, 2021, Montreal, Canada. As we believe that lifelong learning dialogue systems is a very important topic for the future of dialogue systems and AI, we decided to develop the tutorial into a book. Our original interest in the topic stemmed from our research in lifelong/continual machine learning, dialogue systems, and natural language processing. Over the years, we have used many dialogue systems in smartphones and customer support websites. Our experiences have mostly been less than satisfactory. It is very natural to ask the question of why the dialogue system cannot communicate with users and learn to improve itself when it cannot understand what the user says as we humans do. Most of the deployed dialogue systems work in multi-user environments, e.g., Amazon Alexa, Apple Siri, and Google Assistant. If these systems can learn even a tiny amount of knowledge from each user when they get stuck, they will become very knowledgeable and smart over time. Another reason for our interest in lifelong learning dialogue systems is that

our research group has been working on lifelong/continual learning for many years. This combination of factors encouraged us to work on the topic and to write this book.

As lifelong learning dialogue systems sit at the intersection of dialogue systems in natural language processing and lifelong/continual learning in machine learning, this book will touch both fields. We aim to present a comprehensive survey and review of the important research results and latest ideas in these areas. We also want to propose a theoretical framework to be used to guide future research and development in the field. This framework is called SOLA (*Self-initiated Open-world continual Learning and Adaptation*), which was originally proposed for building autonomous and continual learning AI agents. Since lifelong learning dialogue systems are such agents, the framework is naturally suited for the topic. Presently, there are several research topics in dialogue systems that are closely related to lifelong learning dialogue systems. This book will bring all these topics under one roof and discuss their similarities as well as differences. Through this book, we would also like to motivate and encourage researchers to work on and practitioners to build lifelong learning dialogue systems that can be deployed for practical use to improve user experiences and to make dialogue systems smarter and smarter over time. Without the capability of continually learning, accumulating knowledge, making inference about it, and using the knowledge to help future learning and problem solving, achieving true intelligence for AI agents is unlikely.

Two principles have guided the writing of this book. First, it should contain strong motivations for conducting research in lifelong learning dialogue systems in order to encourage graduate students and researchers to work on the problem. Second, the writing should be accessible to practitioners and upper-level undergraduate students who have basic knowledge of natural language processing and machine learning. Yet there should be sufficient in-depth materials for graduate students who plan to pursue Ph.D. degrees in dialogue systems, lifelong learning, or their integration of lifelong learning dialogue systems. We also strongly believe that lifelong learning dialogue systems can be built and deployed for practical applications.

This book is suitable for students, researchers, and practitioners who are interested in dialogue systems, natural language processing, and machine learning. Lecturers can readily use the book in class for courses in any of these related fields.

Santa Clara, USA
Chicago, USA
June 2023

Sahisnu Mazumder
Bing Liu

Acknowledgments

We would like to thank the current and former graduate students in our group and our collaborators: Joydeep Biswas, Jiahua Chen, Zhiyuan Chen, Daniel A. DeLaurentis, Sepideh Esmailpour, Neta Ezer, Geli Fei, Hasan Ghadialy, Yiduo Guo, Scott Grigsby, Estevam R. Hruschka Jr., Wenpeng Hu, Minlie Huang, Reid Hyland, Adam Kaufman, Zixuan Ke, Gyuhak Kim, Tatsuya Konishi, Mori Kurokawa, Huayi Li, Jian Li, Tianrui Li, Yanni Li, Haowei Lin, Zhou Lin, Lifeng Liu, Qian Liu, Guangyi Lv, Chuhe Mei, Piero Molino, Chihiro Ono, Arjun Mukherjee, Nianzu Ma, Shaoshuai Mou, Alexander Politowicz, Qi Qin, Steven Rizzi, Eric Robertson, Gokhan Tur, Vipin Vijayan, Lei Shu, Hao Wang, Mengyu Wang, Yijia Shao, Shuai Wang, Hu Xu, Yueshen Xu, Rui Yan, Yan Yang, Tim Yin, Tim Yuan, Philip S. Yu, Lei Zhang, Dongyan Zhao, Hao Zhou, and Xiaoyan Zhu for their contributions of numerous research ideas and helpful discussions over the years.

We are especially grateful to the anonymous reviewer who read the draft manuscripts very carefully and gave us so many excellent comments and suggestions, which have helped us improve the book tremendously. This book was also benefited significantly from numerous discussions in DARPA SAIL-ON Program meetings.

On the publication side, we thank the editor of Synthesis Lectures on Natural Language Processing, Graeme Hirst, for his valuable guidance in the project. The President and CEO of Morgan & Claypool Publishers, Michael Morgan, and his staff Christine Kiilerich, Springer manuscript editor Susanne Filler, and production editor Bharath Kumar Dhamodharan have given us all kinds of help promptly whenever requested, for which we are very grateful.

Our greatest gratitude goes to our own families. Sahisnu Mazumder would like to thank his parents Ramesh Ch. Mazumder and Dulali Mazumder, wife Mohana, elder sister Snigdha, brother-in-law Sudip, and (late) aunt Malaya Goswami for their invaluable support and encouragements. Bing Liu would like to thank his wife Yue, children Shelley and Kate, and parents Yuhan Liu and Xianzhen Dong. They have helped in so many ways.

The writing of this book was partially supported by a SAIL-ON Program Contract (HR001120C0023) of the Defense Advanced Research Projects Agency (DARPA), Northrop Grumman research gifts, three National Science Foundation (NSF) grants (1650900, IIS-1910424 and IIS-1838770), an NCI grant R01CA192240, and a Research

Contract with KDDI. The content of the book is solely the responsibility of the authors and does not necessarily represent the official views of DARPA, Northrop Grumman, NSF, NCI, KDDI, UIC, or Intel.

The Department of Computer Science at the University of Illinois at Chicago provided computing resources and a very supportive environment for this project. We would particularly like to thank Patricia Brianne Barrera, Sheri Lyn Joscelyn, Denise Marie Kelly, Emily Lam, Sherice Nelson, and Ivy Yuan for their valuable support in the research projects related to this book.

Sahisnu Mazumder would also like to thank Intel Corporation, his colleagues at Multimodal Dialogue and Interaction (MDI) group and Intelligent Systems Research (ISR) division at Intel Labs, and, specifically, Saurav Sahay and Lama Nachman for their invaluable support, advice, and encouragement.

June 2023

Sahisnu Mazumder
Bing Liu

Contents

1 Introduction	1
1.1 Dialogue and Interactive Systems: Background	2
1.1.1 Task-Oriented Dialogue Systems	3
1.1.2 Chit-chat Systems	5
1.2 Weaknesses of Modern Dialogue Systems	7
1.3 Motivation for Lifelong Learning Dialogue Systems	9
1.4 Lifelong Interactive Learning in Conversation	11
1.5 Organization of the Book	13
References	14
2 Open-World Continual Learning: A Framework	21
2.1 Classical Machine Learning	22
2.2 A Motivating Example for SOLA	23
2.3 Novelty Detection	24
2.4 Lifelong and Continual Learning	26
2.5 The SOLA Framework	31
2.5.1 Components of SOLA	32
2.5.2 Open World Continual Learning	34
2.5.3 Relevance of Novelty	36
2.5.4 Novelty Characterization and Adaptation	37
2.5.5 Risk Assessment and Learning	38
2.6 A Dialogue System based on SOLA	39
2.7 Comparison with Related Learning Paradigm	41
2.7.1 Novelty Detection	41
2.7.2 Open-World Learning	41
2.7.3 Continual Learning	42
2.8 Summary	42
References	43

3	Continuous Factual Knowledge Learning in Dialogues	49
3.1	Opportunities for Knowledge Learning in Dialogues	50
3.2	Extracting Facts from Dialogue Context	52
3.2.1	Learning Knowledge Graph from Dialogue	52
3.2.2	Knowledge Acquisition in a Rule-Based Dialogue System	55
3.3	Lexical Knowledge Acquisition in Dialogues	58
3.3.1	Generating Questions for Acquiring Attributes of Unknown Concepts	59
3.3.2	Knowledge Acquisition via Implicit Confirmations over Dialogues	61
3.4	Interactive Factual Knowledge Learning and Inference	63
3.4.1	The CILK Framework	64
3.4.2	The IKAI Framework	67
3.5	Learning New Knowledge from External Sources	69
3.5.1	NELL Architecture	70
3.5.2	Extractors and Learning in NELL	71
3.6	Summary	72
	References	72
4	Continuous and Interactive Language Learning and Grounding	77
4.1	Modes of Language Learning in Human-Chatbot Interactions	79
4.2	Learning Language through Interactions and Demonstration	80
4.2.1	Learning Language Games through Interaction	80
4.2.2	Language Learning through Collaborative Instruction Following	84
4.3	Dialogue-Driven Learning for Adaptive NLI	86
4.3.1	Interactive Teaching of New Concepts for Adaptive NLI	86
4.3.2	Learning Adaptive NLI via Command Matching and Learning	88
4.4	Interactive Semantic Parsing and Learning from Feedback	92
4.4.1	Improving Semantic Parsing with User Interaction	92
4.4.2	Learning Semantic Parser from User Interaction	93
4.5	Summary	95
	References	96
5	Continual Learning in Chit-Chat Systems	103
5.1	Predicting User Satisfaction	104
5.1.1	User Satisfaction Prediction Approaches	105
5.1.2	Dissatisfaction Categorization and Classification	107
5.2	Learning by Extracting New Examples from Conversation	109
5.3	Dialogue Learning via Role-Playing Games	118

5.4 Summary	121
References	122
6 Continual Learning for Task-Oriented Dialogue Systems	127
6.1 Open Intent Detection and Learning	128
6.1.1 Open Intent Detection with Margin Loss	129
6.1.2 Learning Adaptive Decision Boundary	131
6.2 Continual Learning for Semantic Slot Filling	133
6.3 Continual Learning for Dialogue State Tracking	136
6.3.1 Domain-Lifelong Learning for DST	136
6.3.2 Continual Prompt Tuning for DST	138
6.4 Continual Learning for Natural Language Generation	140
6.4.1 Adaptively Regularized Prioritized Exemplar Replay (ARPER)	140
6.4.2 Continual Learning with Transformer Calibration	141
6.5 Joint Continual Learning of All Dialogue Tasks	142
6.5.1 End-to-End Continual Learning for ToDS with AdapterCL	143
6.5.2 SL-AGENT	147
6.6 Summary	147
References	148
7 Continual Learning of Conversational Skills	153
7.1 Learning User Behaviors and Preferences	154
7.2 Learning Emotions, Moods and Opinions in Dialogues	157
7.3 Modeling Situation-Aware Conversations	158
7.4 Summary	159
References	160
8 Conclusion and Future Directions	165
References	171

About the Authors

Sahisnu Mazumder is an AI Research Scientist at Intel Labs, Santa Clara, CA, USA, where he works on Human-AI Collaboration and Dialogue & Interaction Systems research. His research interests include Lifelong and Continual Learning, Dialogue and Interactive Systems, Open-world AI/Learning, Knowledge Base Reasoning, and Sentiment Analysis. He obtained his Ph.D. in Computer Science at the University of Illinois at Chicago, USA, and his Masters in Computer Science from the Indian Institute of Technology (IIT)—Roorkee, India. During his Ph.D., he worked as a Research Intern at Huawei Research, USA, on projects related to user activity and interest mining and at Microsoft Research—Redmond on Natural Language Interaction (NLI) system design. He has published several research papers in leading AI, NLP, and Dialogue conferences like AAAI, IJCAI, ACL, EMNLP, NAACL, SIGDIAL, and CIKM and delivered tutorials in SIGIR—2022, IJCAI—2021, and Big Data Analytics (BDA—2014). He has also served as a Session Chair in COLING 2022 and PC Member/Reviewer of premier conferences like AAAI, IJCAI, ACL, EMNLP, NAACL, EACL, and COLING and journals like ACM TALLIP, Computational Linguistics, and IEEE TNNLS.

Bing Liu is a Distinguished Professor of Computer Science at the University of Illinois at Chicago, USA. He received his Ph.D. in Artificial Intelligence from the University of Edinburgh, UK. His current research interests include Lifelong and Continual Learning, Lifelong Learning Dialogue Systems, Open-World Learning, Sentiment Analysis and Opinion Mining, Machine Learning, and Natural Language Processing. His previous interests include Fake Review Detection, Web Mining, and Data Mining. He has published extensively in top conferences and journals in these areas and authored four books: one about Lifelong/Continual Machine Learning, two about Sentiment Analysis, and one about Web Mining. Three of his papers have received Test-of-Time awards and another one received Test-of-Time honorable mention. Some of his works have been widely reported in popular and technology press internationally. He has served as the Chair of ACM SIGKDD from 2013 to 2017, as program chair of many leading data mining conferences including KDD, ICDM, CIKM, WSDM, SDM, and PAKDD, and as

associate editor of many leading journals such as TKDE, TKDD, TWEB, and DMKD. He is the winner of 2018 ACM SIGKDD Innovation Award, and is a Fellow of AAAI, ACM, and IEEE.



Building *dialogue systems* or *conversational agents* capable of conversing with humans in natural language (NL) and understanding human NL instructions is a long-standing goal of AI (Winograd 1972). These systems, also known as *chatbots*, have become the front runner of AI advancement due to wide-spread applications such as assisting customers in buying products, booking tickets, reducing stress, and executing actions like controlling house appliances and reporting weather information. However, the user experiences have not been fully satisfactory so far (Huang et al. 2020; Li et al. 2016a; Schaub et al. 2021). There are many weaknesses with the current research and fielded dialogue systems. One of the major weaknesses is that they do not learn continuously during conversation (i.e., post-deployment) with the user after they are deployed in practice. Building lifelong learning dialogue systems that possess the capability of continuous learning during conversation is the main topic of discussion of this book. This chapter aims to motivate and provide the foundational idea of building such dialogue systems. Note that, we use the term *chatbots* to refer to all kinds of conversational agents, such as dialogue systems, personal assistants, conversational question-answering systems etc., onward.

In the following sections, we first provide some background of modern dialogue systems (Sect. 1.1) and discuss their general weaknesses (Sect. 1.2), which provide the motivations for studying and building lifelong learning dialogue systems (introduced in Sects. 1.3 and 1.4). Finally, we conclude the chapter with a discussion on the organization of this book in Sect. 1.5.

1.1 Dialogue and Interactive Systems: Background

Dialogue systems can broadly be categorized into two main types (Gao et al. 2019; Jurafsky and Martin 2020):

- (1) **Chit-chat systems** (Chen et al. 2017; Li et al. 2016a, c; Mei et al. 2017; Pandey et al. 2018; Serban et al. 2016, 2018; Shang et al. 2015; Shen et al. 2017, 2018; Vinyals and Le 2015; Wu et al. 2017, 2018; Xing et al. 2017; Yao et al. 2015) are chatbots designed to engage users by conducting the chit-chat type of conversation on a wide range of topics *without having a specific goal to complete*. Examples include *Social Chatbots* like ELIZZA (Weizenbaum 1966), PARRY (Colby 1975), ALICE (Wallace 2009), Microsoft XiaoIce (Zhou et al. 2020), AliMe Chat (Qiu et al. 2017), and many systems in Alexa Prize Socialbot Grand Challenge (Khatri et al. 2018). Such chatbots are built with the goal of supporting seamless conversation with users, and helping them with useful recommendations and mental supports.
- (2) **Task-oriented chatbots** (Bordes et al. 2017; Budzianowski et al. 2018; Lowe et al. 2017; Luo et al. 2019; Shah et al. 2018; Wen et al. 2017a, b; Williams and Young 2007; Zhao et al. 2017) are chatbots designed to assist users to complete tasks based on users' requests, e.g., providing the requested information and taking actions. Most of the popular personal assistants such as Alexa, Siri, Google Home, and Cortana, are task-oriented chatbots. Besides, these types of chatbots are also built as QA Bots to support Question-answering (QA) over knowledge bases, conversational recommendation systems for online product or service recommendations to end-users and as Natural Language Interaction (NLI) systems to enable natural language (NL) driven task completion.

Although the broad goal of task-oriented chatbots is to perform actions or tasks on users' behalf, they can also be of two types based on their design and the nature of interaction with users whom they are meant to support: **(1)** Systems that achieve task completion through *multi-turn dialogues* with users where users express their intents and refine them (based on the feedback from the chatbot) over a sequence of dialogue turns with the system. They are formally known as **Task-oriented Dialogue Systems (ToDS)**. **(2)** The other kind of systems that intend to accomplish tasks through a *single-turn dialogue* where the user provides a NL instruction (command) and the system's goal is to just interpret it by translating it into some actions to be executed by the underlying application. Such systems are formally known as **Natural Language Interfaces (NLIs)**. NLIs are also sometimes referred to as Natural Language Interaction systems in general. Although NLIs are mostly built to support a single-turn interaction with user per task completion goal, they can engage in multi-turn dialogues with user as well (similar to traditional ToDS systems) to better understand the NL instruction and resolve ambiguities (if any) to serve the user better.

Recently, some works have been done to bridge the gap of the two types of chatbots by fusing them through mode switching (Yoshino et al. 2022; Young et al. 2022), which is typically done using a classification model. The system in (Yoshino et al. 2022) also has a module that generates utterances to bridge the two types of chats to ensure a seamless transition.

1.1.1 Task-Oriented Dialogue Systems

A full-fledged Task-oriented Dialogue Systems (ToDS) is mostly designed as a modular system, having six modules, viz., Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), Dialogue State Tracking (DST), Dialogue Policy (DP) Learning, Natural Language Generation (NLG) and Text-to-speech (TTS) Synthesis. Figure 1.1 shows the architecture of a typical ToDS with all the modules integrated with each other. Here, the ASR module is responsible for translating the spoken utterance from the user into text, which is fed to the NLU module for language understanding and the TTS module generates the speech from text which is the NL response generated by the NLG module. The DST and DP modules are often unified and referred to as Dialogue Manager (DM) that is responsible for the progression of the dialogue by managing the dialogue turns.

Often, ASR and TTS modules are studied as a separate field of research and existing works mostly focus on the remaining four components and their interactions in a typical ToDS framework. In particular, they assume ASR and TTS are available to use and deal with only text-based user input and text-based generated output by the rest four modules. Thus, in the rest of the book, we mainly focus on NLU, DST, DP and NLG as the four main modules of a typical ToDS system.

- (1) **Natural Language Understanding (NLU):** The goal of the NLU module is to identify the user intents and extract associated information (slots) from the user utterance. In general, NLU involves solving three subtasks:

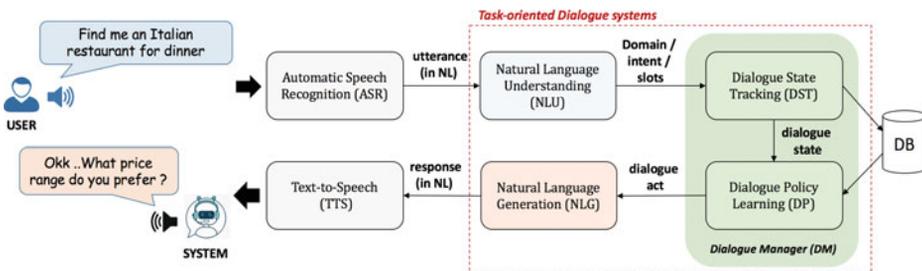


Fig. 1.1 A typical task-oriented dialogue system with its modules

- **Domain Classification**, i.e., classifying the domain of the task expressed in user’s utterance (e.g., is the user talking about airlines, movie, or music?)
- **Intent Classification**, i.e., predicting the task or goal that the user intends to accomplish through the utterance (e.g., is the user wanting to search for a flight, book a movie ticket, or play a song, etc.)
- **Slot filling**, where the goal is to extract the slots and fillers that the user intends the system to understand from his/her utterance given the intent. The problem of Slot filling is commonly formulated as a *sequence labelling task*, where the goal is to tag each word of the utterance with one of the pre-defined semantic slot names.

Let’s have an example here. Considering the user utterance- “*Find me an italian restaurant for dinner*”, the NLU module will classify the domain as “*restaurant*” and intent as “*SearchRestaurant*” and extract slots as {*CUISINE: italian, TIME: dinner*}.

- (2) **Dialogue State Tracking (DST)**: The DST module tracks the dialogue state that captures all the essential information in the conversation so far. Specifically the **dialogue state** includes the entire state of the frame at this point (the fillers of each slot), as well as the user’s most recent dialogue act, summarizing all of the user’s constraints. Below we show an example dialogue snippet, with the dialogue states tracked for each dialogue turn.

User: Find me an Italian restaurant for dinner.

`inform(cuisine=Italian; time=dinner)`

System: Okk. What price range do you prefer?

User: may be in moderate

`inform(cuisine=Italian; time=dinner; price_range=$$)`

System: Sure. There is a nice Italian restaurant at moderate price range nearby.

User: Awesome! What’s the name and address?

`inform(cuisine=Italian; time=dinner; price_range=$$);
request(name; address)`

System: The restaurant name is *Franco’s Ristorante* and address is
300 W 31st St, Chicago, IL 60616.

- (3) **Dialogue Policy (DP) Learning**: The DP Learning module selects the next action based on the current state (obtained from DST), i.e., deciding what action the system should take next, meaning what dialogue act to generate. That is, at a given dialogue turn i , DP predicts which action $\hat{a}_i \in \mathcal{A}$ to take based on the entire dialogue state $S_i = (u_1, a_1, \dots, u_{i-1}, a_{i-1}, u_i)$ [entire sequence of dialogue acts from the system (a) and from the user (u)]:

$$\hat{a}_i = \arg \max_{a_j \in \mathcal{A}} P(a_j | S_i) \quad (1.1)$$

For example, considering the first turn in the above dialogue snippet, with the captured dialogue state as `inform(cuisine=Italian; time=dinner)`, the dialogue act generated by

dialogue policy is `request(price_range)`. This results in the system's response where it asks the user for the price range in the dialogue turn.

`inform(cuisine=Italian; time=dinner)` [*dialogue state*]
 ↓
`request(price_range)` [*dialogue act*]

- (4) **Natural Language Generation (NLG)**: Once the policy has decided what speech act to generate (i.e., \hat{a}_i) for the current dialogue turn, the NLG module generates the text response to the user. It is often modeled in two stages, *content planning* (i.e., what to say) and *sentence realization* (i.e., how to say it). Sentence realization is commonly achieved through *delexicalization* (i.e. mapping from frames to delexicalized sentences by often using encoder-decoder models). Here, we show an example corresponding to the last dialogue turn in the above dialogue snippet, where the system generates the delexicalized response for the given dialogue act `inform(name; address)` as shown below and then, transforms it into an actual NL response by filling appropriate slots in the delexicalized response.

`inform(name; address)` [*dialogue act*]
 ↓
 The restaurant name is `NAME_SLOT` and address is `ADDRESS_SLOT`
 [*delexicalized response*]

Although most of the existing ToDS systems are built as a modularized system, recent approaches have focused on training all these modules together end-to-end to reduce error propagation across modules.

1.1.2 Chit-chat Systems

Chit-chat Systems (or social chatbots) are often implemented as a unitary (non-modular) system. Based on their design approaches, they can broadly be categorized into two types:

- (1) **Rule-based systems**: Rule-based systems work based on patterns and transform rules. Examples include ELIZA (Weizenbaum 1966) and PARRY (Colby et al. 1971). Each pattern/rule is linked to a keyword that might occur in a user utterance and triggers some transformation of a predefined template to generate response. For example, a pattern/transform rule in ELIZA is the following:

(0 YOU 0 ME)	[pattern / decomposition rule]
↓	
(WHAT MAKES YOU THINK I 3 YOU)	[transform / reassembly rule]

Here, “0” in the pattern represents a kleen star (i.e., an indefinite number of words) and “3” in the transform indicates that the third component of the subject decomposition is to be inserted in its place. An example of this pattern/transform rule applied in a dialogue turn would be,

User: You *dislike* me

System: WHAT MAKES YOU THINK I *DISLIKE* YOU

(2) **Corpus-based systems:** Corpus-based systems are trained to mimic human conversations by training on large amounts of human-human conversational data. Examples of this kind of system include modern chatbots like Microsoft XiaoIce (Zhou et al. 2020), AliMe Chat (Qiu et al. 2017) etc. Corpus-based systems are further categorized based on their response production methods as:

- **Response by retrieval** (Bartl and Spanakis 2017; Chen et al. 2017; Lowe et al. 2015; Tao et al. 2019a, b; Wang et al. 2013; Wu et al. 2016, 2017; Yan et al. 2016; Yang et al. 2018; Zhou et al. 2016): These methods formulate the response production task as an information retrieval (IR) problem. Considering the user’s turn as a query q , these methods aim to retrieve and repeat some appropriate turn \hat{r} as the response from a corpus of conversations D (training set for the system). The turns in D form a candidate set of responses, where each turn $r \in D$ is scored as a potential response with respect to the context q . The highest scored candidate is then selected as the response.

$$response(q, D) = \hat{r} = \arg \max_{r \in D} \frac{q \cdot r}{|q||r|} \quad (1.2)$$

- **Response by Generation** (Chen et al. 2017; Li et al. 2016a, c; Mei et al. 2017; Pandey et al. 2018; Serban et al. 2016, 2018; Shang et al. 2015; Shen et al. 2017, 2018; Vinyals and Le 2015; Wu et al. 2018; Xing et al. 2017; Yao et al. 2015): These methods view response production as an encoder-decoder task—transducing from the user’s prior turn to the system’s turn. Figure 1.2 shows an example of response generation by a typical Seq2seq model (Vinyals and Le 2015), for a given conversation context. Here, the encoder is a Recurrent Neural Network (RNN) that encodes the conversation context into a fixed-sized *context vector* to summarize the information in the context and then, another RNN model takes the context vector as input and generates the response word-by-word.

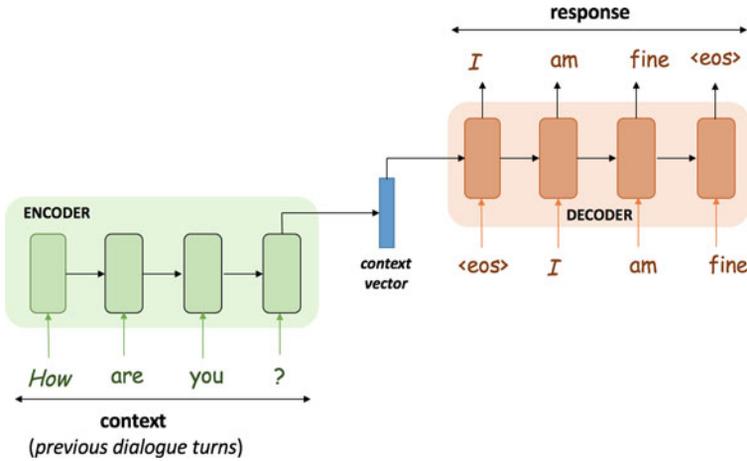


Fig. 1.2 Response generation by a typical seq2seq model

Although there have been a large number of works that solely focus on building a retrieval-based or generative chatbots as discussed above, there are approaches (Qiu et al. 2017; Serban et al. 2017; Song et al. 2016) that adopt a hybrid strategy, i.e., combining neural generative and retrieval based models, which have been shown to achieve better performance. Additionally, Li et al. (2022) attempted to integrate prior knowledge for knowledge-grounded dialogue generation. This method first retrieves relevant information from several knowledge sources (e.g., wiki, dictionary, and knowledge graph), and then transforms the retrieved knowledge into text and concatenates it with the dialogue history to feed into a language model for response generation. Dinan et al. (2019) and Gopalakrishnan et al. (2019) created two knowledge-grounded conversation datasets and trained some state-of-the-art conversation models on them. Zhou et al. (2022) further exploited commonsense knowledge in ConceptNet Speer et al. (2017) in conversation generation. Xu et al. (2022) released a human-human conversation dataset for long conversations and studied the performance of two long-context conversational architectures.

1.2 Weaknesses of Modern Dialogue Systems

Despite their widespread applications, chatbots still have some serious weaknesses:

- (1) **Dull response:** Existing social chatbots often generate uninformative responses like “*I don’t know*”, “*I don’t have a clue*” etc., unspecific responses like “*I like all kinds of foods*” and repeated responses (i.e., telling the user the same thing over and over again) due to lack of diversity (Jiang and de Rijke 2018; Li et al. 2016a, c; See et al. 2019; Zhang et al. 2018). Such dull or uninformative responses often disengage user from the

conversation and leads to the abrupt end of the dialogue session. As we will discuss in Chap. 2, such responses may also occur when the chatbot cannot detect novelty (what it does not understand) well and/or cannot characterize the novelty. As a result, it cannot give a suitable response. In fact, when such a situation occurs, it is opportunity for the chatbot to learn new knowledge.

- (2) **Out of context response:** Whenever the chatbot encounters some *unknown* topics or information in the user’s utterance or *changes in the topic of discussion*, they often go out of context by generating off the track or irrelevant responses, dissatisfying the user. Like the case above, this is also the issue of not being able to detect novelty well and not being able to accommodate or adapt to the novelty, which we will discuss in Chap. 2.
- (3) **Semantic Conflicts:** Open-domain chatbots often suffer from semantic conflict issues in conversation. Here is an example from (Huang et al. 2020):

User: I come from Shanghai.
 System: Cool! I am from Beijing, we come from the same city!
 User: Back to Beijing now. It is so cold but I am very happy!
 System: When will you come to Beijing?

Here, although the user has responded that he/she is currently in Beijing, the chatbot asks when he intends to visit Beijing, creating a semantic conflict in the conversation due to lack of understanding/reasoning over the context.

- (4) **Inconsistency:** Both open-domain and task-oriented chatbots often suffer from an inconsistency problem. Here is an example from (Huang et al. 2020) showing the lack of consistency in open-domain chatbots due to the lack of a coherent personality:

User: Are you a boy or a girl?
 System: I am a boy.
 User: Are you a girl?
 System: Yes, I am a girl.

Schaub et al. (2021) also compiled a list of inconsistent behaviors often observed in modern task-oriented dialogue systems. These include misunderstanding of Intent, Bad entity, Repetition, “Bye” misunderstood, User insults, Bad API call, Incompleteness and Negation error.

- (5) **Lack of context-awareness:** Due to the lack of context-rich datasets used in building chatbots, the deployed chatbots often lack context-awareness (e.g. understanding the situation of the user, dealing with ambiguous utterances from the user that need reference resolution utilizing dialogue history, leveraging multi-model context inputs to generate responses, etc.) while modeling their responses. This sometimes can lead to undesirable outcomes and dissatisfaction in users.
- (6) **Knowledge limitation due to fixed-corpus training:** A great deal of manual effort is needed to label training data or to write rules and compile knowledge bases (KBs). No

matter how much data is used to train a chatbot, it is impossible to cover all possible variations of natural language. Thus, when deployed in practice, a well-trained chatbot often performs poorly. The pre-compiled KBs cannot cover the rich knowledge needed in practice.

A substantial amount of work has focused on solving the above weaknesses over the years to come up with better response generation model that strives to produce more relevant and quality responses. In this book, we focus on presenting and discussing methods that deal with weakness 6 which is the primary motivation for developing continual learning dialogue systems (as discussed next). Existing methods that deal with weakness 6 mainly use fixed knowledge sources and they do not learn new knowledge after deployment (Dinan et al. 2019; Gopalakrishnan et al. 2019; Zhou et al. 2022). Although here we primarily focus on weakness 6, solving this issue has an indirect influence on resolving other issues [1-5] as well, as this provides the chatbot the scope to learn continuously from users and self-improve its response generation capability over time.

1.3 Motivation for Lifelong Learning Dialogue Systems

As we can see, current dialogue systems still have many limitations. Most of us likely have had frustrating experiences engaging with dialogue systems either in smartphones or for customer service. We likely often wish we were instead served in these situations by real humans. Many researchers (including our own research group) have tried to address some of these issues in both task-oriented and chit-chat dialogue systems (Hu et al. 2019a; Shu et al. 2019a, b; Zhou et al. 2018a). In this book, we focus on only the lifelong or continual learning aspect.

The need for lifelong learning dialogue systems is a *no-brainer* simply because that is what we humans do and the current dialogue systems cannot do. In fact, lifelong (or continual) learning is *absolutely necessary* for any truly intelligent system. Regardless of whether an existing dialogue system is based on hand-crafted rules or based on training via a deep learning model, it is an isolated system and can only be improved by human engineers either through adding more labeled training data or hand-crafted rules, or through improving the underlying deep learning models. After any such change, retraining of the whole system is typically required to update the system. This process is highly time consuming as training a deep learning model of a large system usually take days and weeks. However, this is not what we humans do. We humans have this remarkable ability to learn incrementally and quickly and we can learn by ourselves based on our own initiatives through our interactions with those who converse with us and the real-life environment, and learn from books and web pages without being supervised or guided by others. A dialogue system should do the same in order to achieve some level of true intelligence.

Let us use various lifelong/continual learning settings to further motivate why dialogue systems should learn continually. Continual learning in machine learning is defined as learning a sequence of tasks incrementally (Chen and Liu 2018; Ke and Liu 2022; van de Ven and Tolia 2019). In supervised learning, each task is a classification problem that consists of a set of classes to be learned based on its training data. In dialogue systems, it can be a set of new skills that needs to be learned, e.g., booking a hotel and reserving a restaurant table. Continual learning can be further subdivided into two major settings, *class continual learning* (CCL) (also called *class incremental learning*) and *task continual learning* (TCL) (also called *task incremental learning*). In CCL, the classes in each task are distinct (no overlapping of classes with other tasks). At test time, no task related information is provided. For example, the user may say “can you find a near-by Sushi restaurant for me.” The system needs to first predict whether the user wants to find a restaurant or a hotel and then find some near-by Sushi restaurants and give to the user. To be able to incrementally add new skills is clearly important for a dialogue system because retraining the whole system from scratch whenever a new skill is added is very time consuming.

TCL differs from CCL in that TCL allows the user to indicate the task that he/she intends to perform. For example, he/she can select the option of using the restaurant subsystem. In TCL, each task model is usually built separately, but it can share knowledge with models of the previous tasks for knowledge transfer. For example, booking a hotel room and reserving a table in a restaurant have a lot of similarities and knowing one can help do the other. If the system has learned to book a hotel for the user before, it will be much easier for it to learn to help the user reserve a restaurant table, i.e., requiring much less training data. This kind of knowledge transfer in continual learning is done automatically, i.e., the system decides from which previous tasks to transfer and what knowledge to be transferred (Ke et al. 2020, 2021a). Clearly, this ability to transfer knowledge is highly desirable in building practical dialogue systems because manually labeling/writing a large amount of training dialogues is very costly and highly time-consuming. It is also important to note that this kind of transfer is different from the traditional transfer learning. Traditional transfer learning transfers knowledge from a selected source task to a selected target task and the selection is usually done by humans to ensure that the two tasks are similar and can achieve positive transfer. However, in continual learning, everything is done automatically with no manual involvement.

These two forms of learning based on existing research in continual learning, however, are still offline types of learning, meaning that both the tasks and their training data are provided by human engineers, and learning is done offline rather than on the fly during conversation with users. To achieve even more intelligence, it is important that the system can learn actively during conversation with users based on the initiation of the system itself in its interaction with users, i.e., learning on the job after deployment, from users. After all, we humans learn a great deal from our daily conversations.

Another important aspect that needs continual learning is the learning of personal traits, habits, temperaments, emotional characteristics, and each user’s personal situations so that

the dialogue systems can be made more personalized to suit each individual user. This is especially important for dialogue systems that serve as personal assistants. To learn these types of information, the system needs to learn incrementally through a long history of dialogues. To improve this kind of learning and to learn quickly, the system can borrow knowledge from like-minded users and users with similar personal circumstances. In such cases, continual monitoring and incremental learning and automated knowledge transfer are all critical.

The ultimate goal is for dialogue systems to achieve human-like behaviors and abilities in conversation so that they can continually learn and improve themselves on their own to become more and more knowledgeable and powerful without intervention from human engineers. Continual learning can be carried out either through interactions with users during conversation and/or from other sources such as books and web pages. Achieving this ultimate goal is clearly difficult at the moment, but it is possible to progressively improve the dialogue systems technology to move towards the goal gradually.

1.4 Lifelong Interactive Learning in Conversation

In this section, we dive more into online continual learning during conversation rather than offline continual learning. This is perhaps the most central capability that a lifelong learning dialogue system should have. That is, a dialogue system should not be limited by offline training or pre-compiled knowledge bases (KBs). It should learn online on the fly during conversation continually, which is also called *learning on the job or while working*. The process typically involves interaction with human users and learn to improve themselves in a self-motivated and self-initiated manner (Chen and Liu 2018; Liu 2020; Padmakumar and Mooney 2020). We call this form of learning LINC (**Lifelong Interactive learning in Conversation**) (Liu and Mazumder 2021). LINC focuses on three continuous learning capabilities of chatbots: (1) learning factual knowledge in open-ended and information-seeking conversations, (2) learning to ground new natural language utterances, and (3) learning new conversational skills. Some initial attempts have been made in (Hancock et al. 2019; Lee 2017; Luo et al. 2019; Mazumder et al. 2018, 2019, 2020b). A key idea for solving the LINC problem is to exploit **the wisdom of the crowd** in a multi-user environment (where almost all chatbots operate) to learn new knowledge by actively asking or interacting with the current user and/or other users to enable the chatbot to learn a large amount of knowledge quickly and effectively. In Chap. 2, we will define a more general framework (called SOLA) for on-the-job learning.

LINC can be more formally described as follows. During a conversation, the chatbot creates a new task T_{N+1} on the fly when it wants to learn a piece of knowledge from a user utterance (e.g., extracting an unknown fact), encounters a problem (e.g., unable to understand a user utterance), encounters an unknown intent or slot in task-oriented dialogues, or is unable

to answer a user query due to unknown entities or facts).¹ The problem of discovering the unknowns during conversation to formulate a new learning task is referred to as *novelty detection* or *out-of-distribution (OOD) detection* (Liu 2020), solving which is essential for LINC. Once the new task T_{N+1} is formulated, the chatbot needs to acquire the ground truth training data D_{N+1} for learning. To do this, the chatbot needs to *formulate a dynamic interaction strategy* \mathcal{S} to interact with the user, i.e., to decide what to ask and when to ask the user, and then *execute* \mathcal{S} to acquire the ground truth data. It then incrementally learns task T_{N+1} with only one or a few examples.

Existing approaches to obtaining the training data is through manual labeling or writing, which is both costly and time-consuming. As chatbots typically work in multi-user environments, we can exploit such environments to obtain the ground truth training data interactively during actual conversations. This process incurs no cost.

Apart from traditional supervised learning through training using labeled examples, a dialogue system can learn during conversation in many other ways.

1. Extracting data or information directly from user utterances (or dialogue history), which can be real-world facts, user preferences, etc. For example, while conversing about movies, if the user says “*I watched Forest Gump yesterday. The movie was awesome. Liked Tom Hanks’ performance very much.*”, the chatbot can extract the new fact (*Forest Gump, isa, movie*) and (*Tom Hanks, performed_in, Forest Gump*) (Liu and Mei 2020). Later, the chatbot can use these facts in future conversations while answering questions like “*Who acted in Forest Gump?*” or generating a response to user’s utterance “*I’m feeling bored. Can you recommend a good movie?*”. The chatbot may even ask the user some related questions (Liu and Mei 2020) to obtain more knowledge. For example, after obtaining (*Forest Gump, isa, movie*), the chatbot may ask a property question: “*What is the genre of Forest Gump?*” If the user answers, then another piece of knowledge is learned. Note that the extraction method proposed in (Liu and Mei 2020) is rule-based, which works with rule-based chatbots. Many deployed chatbots in industry are written with handcrafted rules. Additional knowledge/data may be inferred from the acquired knowledge and the existing knowledge bases.

2. Asking the current user when the chatbot (1) doesn’t understand a user utterance, or (2) cannot answer a user query, which forms a new learning task. To obtain the ground truth data, for (1), the agent may ask the current user for clarification, rephrasing, or even demonstration if it is supported (Mazumder et al. 2020b). For (2), it may ask the user for some supporting facts and then infer the query answer (Mazumder et al. 2019, 2020a).

3. Asking other users to obtain the answers when the chatbot could not answer a user query. For example, if a user asks “*What is the capital city of the US?*” and the chatbot is unable to answer or infer now, it can try to find a good opportunity to ask another user later “*Hey, do you happen to know what the capital city of the US is?*” If the user gives the answer “*it’s Washington DC,*” the chatbot acquires the ground truth (a piece of new knowledge) which can be learned and used in future conversations. Note that although the

¹ The knowledge learning tasks created by the chatbot for itself to learn are not the same as the tasks that the end-user wants to perform via the chatbot.

answer cannot help the user who asked the question originally, it may be used in the future when a similar question is asked by another user.

4. Observing user demonstrations. In some cases, the chatbots deployed in practice also come with Graphical User Interfaces (GUIs) or remote control facilities to explicitly control devices apart from controlling them via natural language commands. Examples of such systems include robots performing household tasks like cleaning robots and personal assistants integrated with home appliances like Smart TVs, Smart Lights, Smart Speakers, etc. Considering the user has issued a command and the bot has failed to execute the intended action, the user may perform the intended action via the GUI or remote control. The bot can record the sequence of executed action(s) performed by the user by accessing the underlying application logs and store the executed APIs as the ground truth for the input natural language command. The command along with the invoked APIs can serve as labeled examples for learning the command (Forbes et al. 2015; Wang et al. 2017).

5. Extract ground-truth data from external sources, e.g., online documents or online knowledge bases (Mitchell et al. 2015).

Acquiring knowledge from end-users comes with a shortcoming. That is, the knowledge learned from them can be erroneous. Some users may even purposely fool the system by providing wrong information or knowledge. Since chatbots usually work in a multi-user environment, such issues can be addressed through *cross-verification*. After acquiring a piece of new knowledge (a new command pattern or a fact) in an interaction session, the agent can store these new examples in a unverified knowledge buffer. Next, while interacting with some other users in future sessions to accomplish a related task, it can ask these users to verify the accumulated unverified knowledge. Once verified for K times (by K different random users), the knowledge can be considered as trustworthy and removed from the unverified buffer and used in learning or chatting.

1.5 Organization of the Book

This book surveys and introduces the topic of lifelong or continual learning dialogue systems. Although the body of literature is not particularly large, many related papers are published in a number of conferences and journals. There is also a large number of papers that do not exhibit all the characteristics of a continual learning dialogue system and are somewhat *weakly-related* to the topic. It is thus hard, if not impossible, to cover all of the important and related works in the field. As a result, this book should not be taken as an exhaustive account of everything on this topic. However, we believe, the book provides a fairly broad coverage and presents some of the representative works and sets the necessary foundation for future advancements in this area. Also, this book mainly focuses on the advancement in the topic of continual learning dialogue systems. It does not intend to cover the basics of prerequisite knowledge (e.g., machine learning, dialogue systems, NLP) in depth. Interested readers can follow some of the closely-related books (Chen and Liu 2018; Gao et al. 2019; Goodfellow

et al. 2016; Jokinen and McTear 2009; Jurafsky and Martin 2020; McTear 2020) on these topics for prerequisites.

The rest of book is organized as follows. In Chap. 2, we introduce the framework for open-world continual learning, namely, *Self-initiated Open-world continual Learning and Adaptation* (SOLA), which sets the foundational idea for building a continual learning dialogue system.

In Chap. 3, we discuss various opportunities for *continuous factual knowledge learning in dialogues*, and discuss methods about how a chatbot can learn by extracting factual knowledge from conversation, acquiring lexical knowledge and facts interactively from end-users and learning by extracting knowledge from the Web.

Chapter 4 discusses and presents methods for *continual and interactive language learning* for natural language interfaces (NLI). In particular, we discuss about interactive language learning in games, building self-adaptive NLIs by continuously learning new commands from users and interactive semantic parsing for knowledge base question answering with user feedback to deal with parsing errors and ambiguities.

Chapter 5 focuses on *continual learning in chit-chat dialogue systems*. Here, we discuss approaches to predict user satisfaction in conversation and present methods that use user dissatisfaction as an implicit signal to acquire user feedback and leverage it to continuously improve response generation after model deployment.

In Chap. 6, we discuss methods for building *continual learning task-oriented dialogue systems* (ToDS). The chapter discusses the topic of open-world intent learning and presents recent continual learning approaches on various ToDS sub-tasks like slot filling, dialogue state tracking, natural language generation, and also, an approach that attempts to jointly solve all these sub-tasks in an end-to-end continual learning setting.

Chapter 7 provides the motivation for *continual learning of various conversational skills* like personalized conversation modelling, learning of emotions, moods, and opinions in conversation and situation-aware conversation modeling. To the best of our knowledge, we have not come across any work in this topic that is particularly related to continual learning. Thus, here we provide a brief survey of existing works and describe an outline for future research in the topic.

Finally, Chap. 8 concludes the book and discusses some challenges and future directions of research.

References

- A. Bartl, G. Spanakis, A retrieval-based dialogue system utilizing utterance and context embeddings, in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. (IEEE, 2017), pp. 1120–1125
- A. Bordes, Y.-L. Boureau, J. Weston, Learning end-to-end goal-oriented dialog, in *International Conference on Learning Representations* (2017)

- P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, 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*, pp. 5016–5026 (2018)
- Z. Chen, B. Liu, *Lifelong Machine Learning*. (Morgan and Claypool Publishers, 2018)
- H. Chen, X. Liu, D. Yin, J. Tang, A survey on dialogue systems: recent advances and new frontiers. *ACM SIGKDD Explor. Newsl.* **19**(2), 25–35 (2017)
- K.M. Colby, *Artificial Paranoia: a Computer Simulation of Paranoid Processes*, (Elsevier Science Inc., 1975)
- K.M. Colby, S. Weber, F.D. Hilf, Artificial paranoia. *Artif. Intell.* **2**(1), 1–25 (1971)
- E. Dinan, S. Roller, K. Shuster, A. Fan, M. Auli, J. Weston, Wizard of wikipedia: Knowledge-powered conversational agents, in *International Conference on Learning Representations* (2019)
- M. Forbes, R.P.N. Rao, L. Zettlemoyer, M. Cakmak, Robot programming by demonstration with situated spatial language understanding, in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. (IEEE, 2015), pp. 2014–2020
- J. Gao, M. Galley, L. Li, Neural approaches to conversational AI: question answering, task-oriented dialogues and social chatbots. (Now Foundations and Trends, 2019)
- I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*. (MIT press, 2016)
- K. Gopalakrishnan, B. Hedayatnia, Q. Chen, A. Gottardi, S. Kwatra, A. Venkatesh, R. Gabriel, D. Hakkani-Tür, A.A. AI, Topical-chat: towards knowledge-grounded open-domain conversations, in *INTERSPEECH*, pp. 1891–1895 (2019)
- B. Hancock, A. Bordes, P.-E. Mazare, J. Weston, Learning from dialogue after deployment: feed yourself, chatbot!, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3667–3684 (2019)
- W. Hu, Z. Chan, B. Liu, D. Zhao, J. Ma, R. Yan, Gsn: a graph-structured network for multi-party dialogues, in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-2019)* (2019a)
- M. Huang, X. Zhu, J. Gao, Challenges in building intelligent open-domain dialog systems. *ACM Trans. Inf. Syst. (TOIS)* **38**(3), 1–32 (2020)
- S. Jiang, M. de Rijke, Why are sequence-to-sequence models so dull? understanding the low-diversity problem of chatbots, in *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*, pp. 81–86 (2018)
- K. Jokinen, M. McTear, Spoken dialogue systems. *Synth. Lect. Hum. Lang. Technol.* **2**(1), 1–151 (2009)
- D. Jurafsky, J.H. Martin, *Speech and Language Processing: an Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2020)
- Z. Ke, B. Liu, Continual learning of natural language processing tasks: a survey (2022). [arXiv:2211.12701](https://arxiv.org/abs/2211.12701)
- Z. Ke, B. Liu, X. Huang, Continual learning of a mixed sequence of similar and dissimilar tasks. *Adv. Neural Inf. Process. Syst.* **33**, 18493–18504 (2020)
- Z. Ke, B. Liu, H.X. Nianzu Ma, L. Shu, Achieving forgetting prevention and knowledge transfer in continual learning. *Adv. Neural Inf. Process. Syst.* **34**, 22443–22456 (2021)
- C. Khatri, A. Venkatesh, B. Hedayatnia, R. Gabriel, A. Ram, R. Prasad, Alexa prize-state of the art in conversational AI. *AI Mag.* **39**(3), 40–55 (2018)
- S. Lee, Toward continual learning for conversational agents (2017). [arXiv:1712.09943](https://arxiv.org/abs/1712.09943)
- J. Li, M. Galley, C. Brockett, J. Gao, W.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*, pp. 110–119 (2016a)

- J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, J. Gao, Deep reinforcement learning for dialogue generation, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1192–1202 (2016c)
- Y. Li, B. Peng, Y. Shen, Y. Mao, L. Liden, Z. Yu, J. Gao, Knowledge-grounded dialogue generation with a unified knowledge representation, in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 206–218 (2022)
- B. Liu, C. Mei, Lifelong knowledge learning in rule-based dialogue systems (2020). [arXiv:2011.09811](https://arxiv.org/abs/2011.09811)
- B. Liu, Learning on the job: Online lifelong and continual learning. *Proc. AAAI Conf. Artif. Intell.* **34**, 13544–13549 (2020)
- B. Liu, S. Mazumder, Lifelong and continual learning dialogue systems: learning during conversation. *Proc. AAAI Conf. Artif. Intell.* **35**, 15058–15063 (2021)
- R. Lowe, N. Pow, I.V. Serban, L. Charlin, C.-W. Liu, J. Pineau, Training end-to-end dialogue systems with the ubuntu dialogue corpus. *Dialogue & Discourse* **8**(1), 31–65 (2017)
- R. Lowe, N. Pow, I.V. Serban, J. Pineau, The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems, in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 285–294 (2015)
- L. Luo, W. Huang, Q. Zeng, Z. Nie, X. Sun, Learning personalized end-to-end goal-oriented dialog. *Proc. AAAI Conf. Artif. Intell.* **33**, 6794–6801 (2019)
- S. Mazumder, B. Liu, N. Ma, S. Wang, Continuous and interactive factual knowledge learning in verification dialogues, in *NeurIPS-2020 Workshop on Human And Machine in-the-Loop Evaluation and Learning Strategies* (2020a)
- S. Mazumder, B. Liu, S. Wang, S. Esmailpour, An application-independent approach to building task-oriented chatbots with interactive continual learning, in *NeurIPS-2020 Workshop on Human in the Loop Dialogue Systems* (2020b)
- S. Mazumder, B. Liu, S. Wang, N. Ma, Lifelong and interactive learning of factual knowledge in dialogues, in *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 21–31 (2019)
- S. Mazumder, N. Ma, B. Liu, Towards a continuous knowledge learning engine for chatbots (2018). [arXiv:1802.06024](https://arxiv.org/abs/1802.06024)
- M. McTear, Conversational AI: dialogue systems, conversational agents, and chatbots. *Synth. Lect. Hum. Lang. Technol.* **13**(3), 1–251 (2020)
- H. Mei, M. Bansal, M. Walter, Coherent dialogue with attention-based language models, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31 (2017)
- T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, Never-ending learning, in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-15)* (2015)
- A. Padmakumar, R. Mooney, Dialog as a vehicle for lifelong learning, in *Position Paper Track at the SIGDIAL Special Session on Physically Situated Dialogue (RoboDial 2.0)* (2020)
- G. Pandey, D. Contractor, V. Kumar, S. Joshi, Exemplar encoder-decoder for neural conversation generation, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1329–1338 (2018)
- M. Qiu, F.-L. Li, S. Wang, X. Gao, Y. Chen, W. Zhao, H. Chen, J. Huang, W. Chu, Alime chat: a sequence to sequence and rerank based chatbot engine, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 498–503 (2017)

- L.-P. Schaub, V. Hudecek, D. Stancl, O. Dusek, P. Paroubek, Defining and detecting inconsistent system behavior in task-oriented dialogues, in *Traitement Automatique des Langues Naturelles*, pp. 142–152. ATALA (2021)
- A. See, S. Roller, D. Kiela, J. Weston, What makes a good conversation? how controllable attributes affect human judgments, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 1702–1723 (2019)
- I.V. Serban, R. Lowe, P. Henderson, L. Charlin, J. Pineau, A survey of available corpora for building data-driven dialogue systems: the journal version. *Dialogue & Discourse* **9**(1), 1–49 (2018)
- I.V. Serban, C. Sankar, M. Germain, S. Zhang, Z. Lin, S. Subramanian, T. Kim, M. Pieper, S. Chandar, N.R. Ke et al., A deep reinforcement learning chatbot (2017). [arXiv:1709.02349](https://arxiv.org/abs/1709.02349)
- I.V. Serban, A. Sordoni, Y. Bengio, A. Courville, J. Pineau, Building end-to-end dialogue systems using generative hierarchical neural network models, in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 3776–3783 (2016)
- P. Shah, D. Hakkani-Tur, B. Liu, G. Tür, Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pp. 41–51 (2018)
- L. Shang, Z. Lu, 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*, pp. 1577–1586 (2015)
- X. Shen, H. Su, Y. Li, W. Li, S. Niu, Y. Zhao, A. Aizawa, G. Long, A conditional variational framework for dialog generation, in *55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, (Association for Computational Linguistics (ACL), 2017), pp. 504–509
- X. Shen, H. Su, S. Niu, V. Demberg, Improving variational encoder-decoders in dialogue generation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018)
- L. Shu, P. Molino, M. Namazifar, H. Xu, B. Liu, H. Zheng, G. Tür, Flexibly-structured model for task-oriented dialogues, in *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 178–187 (2019a)
- L. Shu, H. Xu, B. Liu, P. Molino, Modeling multi-action policy for task-oriented dialogues, in *Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP-2019, short paper)* (2019b)
- Y. Song, R. Yan, X. Li, D. Zhao, M. Zhang, Two are better than one: an ensemble of retrieval-and generation-based dialog systems (2016). [arXiv:1610.07149](https://arxiv.org/abs/1610.07149)
- R. Speer, J. Chin, C. Havasi, Conceptnet 5.5: an open multilingual graph of general knowledge, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31 (2017)
- C. Tao, W. Wu, C. Xu, W. Hu, D. Zhao, R. Yan, Multi-representation fusion network for multi-turn response selection in retrieval-based chatbots, in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 267–275 (2019a)
- C. Tao, W. Wu, C. Xu, W. Hu, D. Zhao, R. Yan, One time of interaction may not be enough: go deep with an interaction-over-interaction network for response selection in dialogues, in *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 1–11 (2019b)
- G.M. van de Ven, A.S. Tolias, Three scenarios for continual learning (2019). [arXiv:1904.07734](https://arxiv.org/abs/1904.07734)
- O. Vinyals, Q. Le, A neural conversational model (2015). [arXiv:1506.05869](https://arxiv.org/abs/1506.05869)
- R.S. Wallace, The anatomy of ALICE, in *Parsing the Turing Test*. (Springer, 2009), pp. 181–210
- S.I. Wang, S. Ginn, P. Liang, C.D. Manning, Naturalizing a programming language via interactive learning, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 929–938 (2017)

- H. Wang, Z. Lu, H. Li, E. Chen, A dataset for research on short-text conversations, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 935–945 (2013)
- J. Weizenbaum, Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM* **9**(1), 36–45 (1966)
- T.-H. Wen, Y. Miao, P. Blunsom, S. Young, Latent intention dialogue models, in *International Conference on Machine Learning*. (PMLR, 2017b), pp. 3732–3741
- T.H. Wen, D. Vandyke, N. Mrkšić, M. Gašić, L.M. Rojas-Barahona, P.H. Su, S. Ultes, S. Young, A network-based end-to-end trainable task-oriented dialogue system, in *15th Conference of the European Chapter of the Association for Computational Linguistics*, vol. 1, pp. 438–449 (2017a)
- J.D. Williams, S. Young, Partially observable Markov decision processes for spoken dialog systems. *Comput. Speech & Lang.* **21**(2), 393–422 (2007)
- T. Winograd, Understanding natural language. *Cogn. Psychol.* (1972)
- Y. Wu, W. Wu, Z. Li, M. Zhou, Topic augmented neural network for short text conversation (2016). CoRR [arXiv:1605.00090](https://arxiv.org/abs/1605.00090)
- Y. Wu, W. Wu, C. Xing, M. Zhou, Z. Li, Sequential matching network: a new architecture for multi-turn response selection in retrieval-based chatbots, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 496–505 (2017)
- Y. Wu, W. Wu, D. Yang, C. Xu, Z. Li, Neural response generation with dynamic vocabularies, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018)
- C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, W.-Y. Ma, Topic aware neural response generation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31 (2017)
- J. Xu, A. Szlam, J. Weston, Beyond goldfish memory: long-term open-domain conversation, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5180–5197 (2022a)
- R. Yan, Y. Song, H. Wu, Learning to respond with deep neural networks for retrieval-based human-computer conversation system, in *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 55–64 (2016)
- L. Yang, M. Qiu, C. Qu, J. Guo, Y. Zhang, W.B. Croft, J. Huang, H. Chen, Response ranking with deep matching networks and external knowledge in information-seeking conversation systems, in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 245–254 (2018a)
- K. Yao, G. Zweig, B. Peng, Attention with intention for a neural network conversation model (2015). [arXiv:1510.08565](https://arxiv.org/abs/1510.08565)
- T. Yoshino, Y. Fukuchi, S. Matsumori, M. Imai, Chat, shift and perform: bridging the gap between task-oriented and non-task-oriented dialog systems (2022). [arXiv:2206.11813](https://arxiv.org/abs/2206.11813)
- T. Young, F. Xing, V. Pandealea, J. Ni, E. Cambria, Fusing task-oriented and open-domain dialogues in conversational agents. *Proc. AAAI Conf. Artif. Intell.* **36**, 11622–11629 (2022)
- Y. Zhang, M. Galley, J. Gao, Z. Gan, X. Li, C. Brockett, B. Dolan, Generating informative and diverse conversational responses via adversarial information maximization. *Adv. Neural Inf. Process. Syst.* **31**, (2018b)
- T. Zhao, A. Lu, K. Lee, 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*, pp. 27–36 (2017)
- X. Zhou, D. Dong, H. Wu, S. Zhao, D. Yu, H. Tian, X. Liu, R. Yan, Multi-view response selection for human-computer conversation, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 372–381 (2016)
- P. Zhou, K. Gopalakrishnan, B. Hedayatnia, S. Kim, J. Pujara, X. Ren, Y. Liu, D. Hakkani-Tur, Think before you speak: explicitly generating implicit commonsense knowledge for response genera-

- tion, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1237–1252 (2022)
- H. Zhou, M. Huang, T. Zhang, X. Zhu, B. Liu, Emotional chatting machine: emotional conversation generation with internal and external memory, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018a)
- L. Zhou, J. Gao, D. Li, H.-Y. Shum, The design and implementation of xiaoice, an empathetic social chatbot. *Comput. Linguist.* **46**(1), 53–93 (2020)



As more and more AI agents are used in practice, we need to think about how to make these agents fully autonomous so that they can **(1)** learn by themselves continually in a *self-motivated* and *self-initiated* manner rather than being retrained offline periodically on the initiation of human engineers and **(2)** accommodate or adapt to unexpected or novel circumstances. As the real-world is an open environment that is full of unknowns or novelties, *detecting* novelties, *characterizing* them, *accommodating or adapting* to them, and *gathering* ground-truth training data and *incrementally learning* the unknowns/novelties are critical to making AI agents more and more knowledgeable and powerful over time. The key challenge is how to automate the process so that it is carried out continually on the agent's *own initiative* and *through its own interactions* with humans, other agents and the environment just like human *on-the-job learning* (Liu and Mazumder 2021; Mazumder 2021).

This chapter develops a theoretical framework for open-world continual learning, which also serves as a framework for lifelong learning dialogue systems because such a dialogue system works in an open environment. Section 2.6 will briefly describe a dialogue system that follows the proposed framework. Since we want the framework to be applicable to AI agents in general, the framework will cover aspects that may not be necessary for a lifelong learning dialogue system. The key aspect of the framework is *self-initiation*, which involves no engineers. The proposed framework is called **Self-initiated Open-world continual Learning and Adaptation (SOLA)** (Liu et al. 2023). SOLA is like human *learning on the job* or *learning while working*. It learns after model deployment.

We should note that the SOLA framework goes beyond the traditional concept of machine learning or continual/lifelong learning, which normally starts with a labeled training dataset given to a machine learning algorithm to produce a model (e.g., a classifier). Thus, existing paradigms of *classical machine learning* and *lifelong/continual learning* are only two aspects of the SOLA paradigm because in order to learn on the job, the system must discover and create new tasks to learn and also acquire labeled ground-truth training data during

application on the fly by the agent itself. Furthermore, the agent must adapt to or accommodate unknowns or novelties. Before discussing the SOLA framework, let us first discuss the classical machine learning paradigm.

2.1 Classical Machine Learning

The current dominant paradigm for machine learning (ML) is to run an ML algorithm on a given dataset to generate a model. The model is then applied on a real-life performance task. This is true for both supervised learning and unsupervised learning. We call this paradigm *closed-world learning* because it makes the *independent and identically distributed* (I.I.D) assumption and it does not consider any other related information or the previously learned knowledge. Intuitively, by closed-world learning (Fei and Liu 2016; Liu 2020), we mean what the agent sees in testing or application have been seen in training (we will discuss this further later).

Figure 2.1 illustrates the *classical isolated learning paradigm*, which consists of two stages: (1) *model building* and (2) *model application* or deployment. In model building, the training data D of task T is used by the **Learner** (a ML algorithm) to produce a **Model**. This process is reflected by the blue links in the figure. In model application or deployment, the input data sensed by the **Sensor** from the application is sent to the **Model**, which produces a decision or action to be executed by the **Executor** in the application environment. This process is reflected by the black links. This classical learning paradigm has several limitations.

(1) Most real-life learning environments do not satisfy the I.I.D assumption because they are often dynamic and open, meaning that there may be constant data distribution changes and unknown objects appearing. In such cases, periodical retraining is needed typically initiated by human engineers. To do retraining, manual labeling of new or additional training data is required, which is very labor-intensive and time-consuming. Since the world is too complex and constantly changing, labeling and retraining need to be done periodically. This is a daunting task for human annotators and engineers. This is also the case for building and maintaining a chatbot. Even for unsupervised learning, collecting a large volume of data constantly may not be possible in many cases.

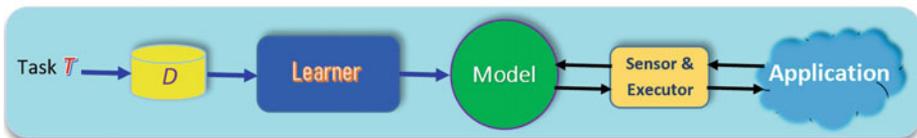


Fig. 2.1 Architecture of the classical machine learning paradigm, where T is the task and D is its training data. The links in blue reflect the learning process and the links in black reflect the application process of the learned model

(2) It learns in isolation, meaning that it does not retain or accumulate knowledge learned in the past and use it to help future learning to achieve knowledge transfer. This is in sharp contrast to our human learning. We humans never learn in isolation. We accumulate and maintain the knowledge learned from previous tasks and use it seamlessly in learning new tasks and solving new problems. That is why whenever we encounter a new situation or problem, we may notice that many aspects of it are not really new because we have seen them in the past in some other contexts. When faced with a new problem or a new environment, we can adapt our past knowledge to deal with the new situation and also learn from it. Over time we learn more and more, and become more and more knowledgeable and more and more effective at learning. In the context of chatbots, it is easy to see opportunities for knowledge transfer. For example, after learning to book a meeting room, learning to reserve a table in a restaurant should be easier.

(3) There is no learning on the job after the model has been deployed (also called post-deployment). Human learning is different as we continue to learn on the job after formal training. Studies have shown that about 70% of human knowledge is learned while working on a task or on the job. Only about 10% is learned through formal training and the rest 20% is learned through imitation of others. An AI system should also learn on the job during model applications. In the context of chatbots, this means to learn during conversation with users.

In the next section, we use a motivating example to illustrate that this classical paradigm is not sufficient in many practical applications and what are involved in building an autonomous learning system that can learn after deployment on the fly. In order to be general, the example is not mainly about dialogue systems but it involves a dialogue system. However, after presenting the SOLA framework, we will briefly discuss a lifelong learning dialogue system that follows the SOLA framework.

2.2 A Motivating Example for SOLA

The example is about a hotel greeting bot (Chen and Liu 2018). It works in an **open environment** full of unknowns or novel objects, which the classical machine learning cannot handle. In general, to make an AI agent such as the greeting bot to thrive in the real open world, like humans, it has to detect novelties and learn them incrementally to make the system more knowledgeable and adaptable over time. It must do so *on its own initiative on the job* (after deployment) rather than relying on human engineers to retrain the system offline periodically. That is, it must learn in the *open world* in a *self-motivated manner* in the context of its *performance task* (the main task of the agent).

Our hotel greeting bot's *performance task* is to greet hotel guests. When its vision system sees a guest (say, John) it has learned before, it greets him by saying,

“Hi John, how are you today?”

When it sees a new guest, it should detect this guest as new or novel. This is a **novelty detection** task (also known as *out-of-distribution (OOD) detection*). Upon discovering the novelty, the new guest, it needs to **accommodate** or **adapt** to the novel situation. The bot may say to the new guest

“Hello, welcome to our hotel! What is your name, sir?”

If the guest replies “David,” the bot takes some pictures of the guest to **gather training data** and then **incrementally or continually learn** to recognize David. The name “David” serves as the *class label* of the pictures taken. Like humans, the detected novelty serves as an intrinsic *self-motivation* for the agent/bot to *learn*. When the bot sees this guest again next time, it may say

“Hi David, how are you today?” (David is no longer novel)

In an actual hotel, the situation is, however, much more complex than this. For example, how does the system know that the novel object is actually a person, not a dog? Even if the system can recognize the object as a person, how does it know that he/she is a hotel guest, not a service provider for services such as delivery or security? In order to adapt to the novel object or situation, the system must first **characterize** the novel object, as without it, the agent would not know how to **adapt** or **respond**. In this case, some classification or similarity comparison is needed to decide whether it is a person with luggage. If the object looks like a person but has no luggage, the bot will not respond or learn to recognize the person as it is **irrelevant** to its performance task. If the novel object looks like an animal, it should notify a hotel employee and learn to recognize the object so that it will no longer be novel when it is seen next time. In short, for each *characterization*, there is a corresponding *response* or *adaptation strategy*, which can be NIL (i.e., do nothing). This discussion shows that in order to characterize, the agent must already have a rich volume of world knowledge. Last but not least, there is also **risk** involved when making an incorrect decision.

The proposed SOLA framework is exactly for dealing with this complex learning environment. Since novelty detection and continual learning are two key issues in SOLA and extensive research has been done about them in the research community, we review them first.

2.3 Novelty Detection

Novelty detection is also called *out-of-distribution (OOD) detection*, *open-world classification* or *open-set recognition*. It is similar to or the same as the earlier *outlier detection* and *anomaly detection* depending on contexts. Novelty is an agent-specific concept. An object may be novel to one agent based on its partial knowledge of the world but not novel to another agent. We distinguish two types of novelty, *absolute novelty* and *contextual novelty*.

Absolute novelty. Absolute novelty represents something that the agent has never seen before. For example, in the context of supervised learning, the agent’s world knowledge is

learned from the training data $\mathcal{D}_{Tr} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in X$ (input space) and $y_i \in Y_{Tr}$ (label space). Let $h(\mathbf{x})$ be the latent or internal representation of \mathbf{x} in the agent’s mind, $h(D_{Tr}^i)$ be the latent representation of the training data D_{Tr}^i of class y_i , and k ($= |Y_{Tr}|$) be the total number of training classes. We use $\mu(h(\mathbf{x}), h(D_{Tr}^i))$ to denote the novelty score of a test instance \mathbf{x} with respect to $h(D_{Tr}^i)$. The degree of novelty of \mathbf{x} with respect to D_{Tr} , i.e., $\mu(h(\mathbf{x}), h(D_{Tr}))$, is defined as the minimum novelty score with regard to every class,

$$\mu(h(\mathbf{x}), h(D_{Tr})) = \min(\mu(h(\mathbf{x}), h(D_{Tr}^1)), \dots, \mu(h(\mathbf{x}), h(D_{Tr}^k))) \quad (2.1)$$

The novelty function μ can be defined based on specific applications. For example, if the training data of each class follows the Gaussian distribution, one may use the Mahalanobis distance as the novelty score.

Novel instance: A test instance \mathbf{x} is *novel* or *out-of-distribution* (OOD) if its novelty score $\mu(h(\mathbf{x}), h(D_{Tr}))$ is greater than or equal to a threshold value γ such that \mathbf{x} can be assigned a new class that is not in Y_{Tr} .

Novel class: A newly created class y_{new} ($y_{new} \notin Y_{Tr}$) assigned to some novel instances is called a *novel class* (or *out-of-distribution* (OOD), *unknown* or *unseen class*). The classes in Y_{Tr} are called *in-distribution* (IND), *known* or *seen*, classes.

Contextual novelty. Based on the prior knowledge of the agent, the probability $P(\mathbf{x}|Q)$ of \mathbf{x} occurring in a particular context Q is very low, but \mathbf{x} has occurred in Q , which is *surprising* or *unexpected*. Both \mathbf{x} and Q are not absolutely novel as they separately have been seen before. A contextual novelty is also commonly called a *surprise* or *unexpected event*. In human cognition, surprise is an emotional response to an instance which greatly exceeds the expected uncertainty within the context of a task. The definitions of contextual novel instance and class are similar to those for absolute novelty.

Intuitively, in absolute novelty, the novelty of \mathbf{x} is context independent. For example, if the agent has never seen a tiger before, it is an absolute novelty, irrespective of its context (other objects in the image). If \mathbf{x} is known but is very unlikely to appear in a context, \mathbf{x} is contextually novel, for example, a deer (known) appearing in a crowded city street. Related work on contextual novelty detection can be found in Ma et al. (2021), Ma et al. (2022).

Novelty is not restricted to the perceivable physical world but also includes the agent’s internal world, e.g., novel interpretations of world states or internal cognitive states that have no correspondence to any physical world state. Interested readers may also read Boulton et al. (2021) for a more nuanced and perception-based study of novelty.

Outlier and anomaly: An outlier is a data point that is far away from the main data clusters, but it may not be unknown. For example, the salary of a company CEO is an outlier with regard to the salary distribution of the company employees, but it is known and thus not novel. Unknown outliers are novel. Anomalies can be considered as outliers or instances that are one off and never repeated. Though technically “novel” they may not need to result in a new class.

Note that this book does not deal with various types of data shift such as *covariate shift*, *prior probability shift* and *concept drift* as a large amount of work has been done (Moreno-Torres et al. 2012).

We will not discuss novelty detection further because it has been studied extensively in the literature. Several excellent surveys exist (Pang et al. 2021; Parmar et al. 2023; Yang et al. 2021a). Some of the related works in the context of dialogue are (Kim et al. 2020b; Lin and Xu 2019a; Lin et al. 2020; Shu et al. 2021; Yan et al. 2020; Zhang et al. 2021b).

2.4 Lifelong and Continual Learning

Human brains have this extraordinary ability to learn a large number of tasks incrementally with high accuracy. Both the learning process of and the learned knowledge for the tasks have little negative interference of each other. In fact, the learned knowledge earlier can even help the learning of new tasks later. *Continual learning* or *lifelong learning* attempts to make the computer to do the same. The concept of *lifelong learning* (LL) was proposed around 1995 in Thrun and Mitchell (1995). Since then, it has been pursued in several directions, e.g., lifelong supervised learning (Chen and Liu 2018), continual learning in deep neural networks (Chen and Liu 2018; Ke and Liu 2022), lifelong unsupervised learning (Chen and Liu 2014), lifelong semi-supervised learning (Mitchell et al. 2015), and lifelong reinforcement learning Ammar et al. 2015. LL techniques working in other areas also exist. Silver et al. (2013) wrote an excellent survey of early LL approaches. A more complete treatment of LL can be found in Chen and Liu (2018).

The terms *lifelong learning* and *continual learning* have the same meaning and are used interchangeably now, but the past research under the two names has focused on different aspects of the same problem.

The early definition of lifelong learning (LL) in Chen and Liu (2018) is follows, which is based on the early definitions in Thrun and Mitchell (1995), Silver et al. (2013), Ruvolo and Eaton (2013), and Mitchell et al. (2015):

Lifelong learning: At any time point, the learner has learned a sequence of N tasks, $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_N$. When faced with the $(N + 1)^{th}$ task \mathcal{T}_{N+1} , the learner can leverage the knowledge learned in the past in the knowledge base (KB) to help learn \mathcal{T}_{N+1} . KB maintains the knowledge learned from the previous N tasks. After the completion of learning \mathcal{T}_{N+1} , KB is updated with the knowledge gained from learning \mathcal{T}_{N+1} .

We can see the goal of the earlier lifelong learning is to leverage the knowledge learned in the past to learn the new task \mathcal{T}_{N+1} better, i.e., *knowledge transfer*. An implicit assumption of LL is that the tasks learned are very similar (Chen and Liu 2018). The learning setting is almost exclusively the *task continual learning* (TCL) setting (see below), where each task is a separate problem.

Continual learning: The term *continual learning* (CL) is more commonly used than *lifelong learning* in the deep learning community. Although knowledge transfer is also a

goal of CL, the focus of CL has been on solving the *catastrophic forgetting* (CF) problem (Chaudhry et al. 2020; Guo et al. 2022a; Ke et al. 2020; Kim et al. 2022a; Kirkpatrick et al. 2017; Lee et al. 2019; Rebuffi et al. 2017; Rusu et al. 2016; Serra et al. 2018; Shin et al. 2017; Zenke et al. 2017). CF refers to the phenomenon that when a neural network learns a sequence of tasks, the learning of each new task is likely to change the weights or parameters learned for previous tasks, which degrades the model accuracy for the previous tasks (McCloskey and Cohen 1989).

In the past few years, CF has attracted a great deal of research attention (Chen and Liu 2018; Ke and Liu 2022). There are two main setups in continual learning: *class continual learning* (CCL) (also called *class incremental learning* (CIL)) and *task continual learning* (TCL) (also called *task incremental learning* (TIL) (van de Ven and Tolias 2019).

Class continual learning (CCL). In CCL, each task consists of one or more classes to be learned together but only one model is learned to classify all classes that have been learned so far. In testing, a test instance from any class may be presented to the model for it to classify with no task information given. Formally, given a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_N, \mathcal{T}_{N+1}, \dots$ and their corresponding datasets $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_N, \mathcal{D}_{N+1}, \dots$. The dataset of task \mathcal{T}_k is $\mathcal{D}_k = \{(x_k^i, y_k^i)_{i=1}^{n_k}\}$, where n_k is the number of data samples in task k , and $x_k^i \in \mathbf{X}$ is an input sample and $y_k^i \in \mathbf{Y}_k$ is its class label. All \mathbf{Y}_k 's are disjoint and $\bigcup_{k=1}^T \mathbf{Y}_k = \mathbf{Y}$, where T is the index of the last task \mathcal{T}_T that has been learned. The goal of CCL is to learn a single prediction function or model $f : \mathbf{X} \rightarrow \mathbf{Y}$ that can identify the target class y for a given test instance x .

Task continual learning (TCL). In TCL, each task is a separate classification problem (e.g., one classifying different breeds of dogs and one classifying different types of birds). TCL builds a set of classification models (one per task) in one neural network. In testing, the system knows which task each test instance belongs to and uses only the model for the task to classify the test instance. Note that the classical LL mainly works in this TCL setting and assumes that the tasks are similar to each other to enable knowledge transfer across tasks. Formally, given a sequence of tasks $\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_N, \mathcal{T}_{N+1}, \dots$ and their corresponding datasets $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_N, \mathcal{D}_{N+1}, \dots$. Each task \mathcal{T}_k has a training dataset $\mathcal{D}_k = \{(x_k^i, k), y_k^i\}_{i=1}^{n_k}$, where n_k is the number of data samples in task $\mathcal{T}_k \in \mathbf{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_T\}$, where \mathcal{T}_T is the last task that has been learned, and $x_k^i \in \mathbf{X}$ is an input sample and $y_k^i \in \mathbf{Y}_k \subset \mathbf{Y}$ is its class label. The goal of TCL is to construct a predictor $f : \mathbf{X} \times \mathbf{T} \rightarrow \mathbf{Y}$ to identify the class label $y \in \mathbf{Y}_k$ for (x, k) (the given test instance x from task k).

From now on, we will only use the term *continual learning* (CL) to mean both lifelong learning and continual learning. CL aims to achieve two main objectives, i.e., (1) overcoming CF and (2) performing cross task knowledge transfer. Clearly, not all problems can achieve both. For example, it is not obvious that different tasks or classes can help each other in CCL except some feature sharing. For TCL, if the tasks are entirely different, it is hard to improve the new task learning via knowledge transfer either. For example, one task is to classify whether one has a heart disease or not but another is to classify whether a loan application should be approved or not. In these cases, CF is the only problem to solve. Recent research

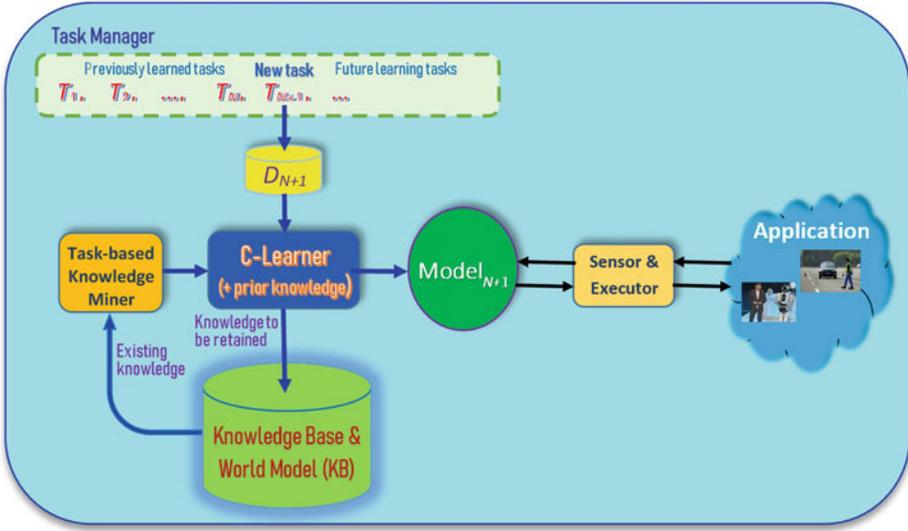


Fig. 2.2 Architecture of a traditional lifelong learning framework. $\mathcal{T}_1, \dots, \mathcal{T}_N$ are the previously learned tasks, \mathcal{T}_{N+1} is the current new task to be learned and D_{N+1} is its training data. The C-Learner (*Continual Learner*) learns by leveraging the relevant prior knowledge identified by the Task-based Knowledge Miner from the Knowledge Base (KB), which contains the retained knowledge learned in the past. It also deals with the catastrophic forgetting

has shown that when a mixed sequence of similar and dissimilar tasks are learned in TCL, it is possible to perform selective knowledge transfer among similar tasks (Ke et al. 2020) and also to overcome CF for dissimilar tasks. Task similarity is detected automatically.

The architecture of CL systems is given in Fig. 2.2. Apart from those standard components and processes of classical machine learning as explained in Sect. 2.1, this new paradigm has some additions and changes.

1. **C-Learner:** C-Learner means *Continual-Learner*. For CL, it is beneficial for the learner to use the prior knowledge in learning the new task. We also call such a learner a *knowledge-based learner*, which can leverage the knowledge in the KB to learn the new task. The knowledge relevant to the current is mined by the *Task Knowledge Miner* (TKM) (see below). C-Learner will also deal with CF in learning a new task.
2. **Knowledge Base and World Model (KB):** It stores the previously learned knowledge and the world model related to the application task. It may also have a few sub-components:
 - (a) *Past Information Store (PIS)*: It stores the information resulted from the past learning, including the resulting models, patterns, or other forms of outcome. As for what information or knowledge should be retained, it depends on the learning task and the

learning algorithm. For a particular system, the user needs to decide what to retain in order to help future learning. For example, in the *experience-replay* based CL approach, a small number of training examples from each previous task is saved so that they can be used to help deal with CF in learning a new task.

- (b) *Knowledge Miner* (MKM). It performs meta-mining of the knowledge in the PIS. We call this *meta-mining* because it mines higher-level knowledge from the saved knowledge. The resulting knowledge is stored in the Meta-Knowledge Store. Here multiple mining algorithms may be used to produce different types of results.
- (c) *Meta-Knowledge Store* (MKS): It stores the knowledge mined or consolidated from PIS (Past Information Store) and also from MKS itself. Some suitable knowledge representation schemes are needed for each application.
- (d) *Knowledge Reasoner* (KR): It makes inference based on the knowledge in MKB and PIS to generate more knowledge. Most current systems do not have this sub-component. However, with the advance of CL, this component will become increasingly important.

Since the current CL research is still in its infancy, none of the existing systems has all these sub-components.

3. **Task-based Knowledge Miner (TKM)**: This module makes use of the raw knowledge or information in the KB to mine or identify knowledge that is appropriate for the current task. This is needed because in many cases, C-Learner cannot directly use (all) the raw knowledge in the KB. For example, the irrelevant knowledge to the current task needs to be filtered out or blocked. And, C-Learner may only be able to use some more general knowledge mined from the KB (Chen and Liu 2014; Ke et al. 2020).
4. **Model**: This is the learned model, which can be a prediction model or classifier in supervised learning, clusters or topics in unsupervised learning, a policy in reinforcement learning, etc. Since the architecture enables continual learning, $Model_{N+1}$ includes all the models from \mathcal{T}_1 to \mathcal{T}_{N+1} , which may all be in one neural network. In the case of TCL, they may be separate models indexed by their task identifiers. In the case of CCL, $Model_{N+1}$ is just one model that covers all classes of the tasks learned so far.
5. **Task Manager (TM)**: It receives and manages the tasks that arrive in the system, and handles the task shift and presents the new learning task to the C-Learner in a lifelong manner.

Continual Learning Process: A typical continual learning process starts with the Task Manager assigning a new task to the C-Learner. C-Learner then works with the help of the past knowledge stored in the KB to produce an output model for the user and also send the information or knowledge that needs to be retained for future use to the KB. Note that dealing with CF is not reflected in the architecture in Fig. 2.2 as it stays in the algorithm of the C-Learner.

Main Continual Learning Approaches. Most recent research papers on CL focus on overcoming CF by protecting what the system has learned previously (Ke and Liu 2022). There are a large number of existing approaches. They can be roughly grouped into three main categories.

Regularization-based approaches: The main idea of these approaches is to compute the importance of each parameter or gradient to the previously learned tasks, and then add a regularization in the loss function to penalize changes to those important parameters to prevent CF on previous tasks (Ahn et al. 2019; Camoriano et al. 2017; Dhar et al. 2019; Jung et al. 2016; Lee et al. 2019; Liu et al. 2020; Schwarz et al. 2018; Zhu et al. 2021). For example, EWC (Kirkpatrick et al. 2017), one of the most popular algorithms, uses the Fisher information matrix to represent the importance of parameters. This technique works to some extent, but is very weak at dealing with CF. SI (Zenke et al. 2017) was proposed to extend EWC so that it has less complexity in computing the penalty. Several approaches (Buzzega et al. 2020; Li and Hoiem 2016) also use knowledge distillation (Hinton et al. 2015) to preserve the previous knowledge.

Replay-based approaches: The idea of these approaches is to use a small memory buffer to save a small amount of data from previous tasks and replay or use them to jointly train the new task together with the new task data to prevent forgetting of the knowledge learned from previous tasks (Buzzega et al. 2020; Cha et al. 2021; Chaudhry et al. 2019; Guo et al. 2022b; Liu et al. 2021b; Lopez-Paz and Ranzato 2017; Rajasegaran et al. 2019; Rebuffi et al. 2017; Rolnick et al. 2019; Rusu et al. 2016). These approaches are also called *experience replay* or *memory-based* approaches. Example systems include GEM (Lopez-Paz and Ranzato 2017) and A-GEM (Chaudhry et al. 2019). Instead of saving some previous data, some approaches learn a data generator for previous tasks (Gepperth and Karaoguz 2016; Hu et al. 2019b; Kamra et al. 2017; Kemker and Kanan 2018; Ostapenko et al. 2019; Seff et al. 2017; Shin et al. 2017). In learning a new task, the generator generates pseudo-samples of previous tasks and uses them instead of real samples to jointly train the new task.

Parameter isolation-based (or architectural approaches): These approaches are mainly used in task continual learning (TCL). Its main idea is to learn a sub-network for each task, and tasks may share some parameters and neurons (Ke et al. 2020; Serra et al. 2018). HAT (Serra et al. 2018) and SupSup (Wortsman et al. 2020) are two representative systems. HAT learns neurons (not parameters) that are important for each task and “hard masks” them via the task embeddings based on gates. SupSup uses a different approach but it also learns and fixes a sub-network for each task. Besides them, many other systems also take similar approaches, e.g., Progressive Networks (Rusu et al. 2016), PathNet (Mallya and Lazebnik 2018), HyperNet (von Oswald et al. 2020, and BNS (Qin et al. 2021).

Kim et al. (2022) and Kim et al. (2023b) have shown that successful class continual learning (CCL) requires both good within-task prediction as well as task-id prediction and that CCL is learnable. They have also proved that task-id prediction is correlated with out-of-distribution (OOD) detection for each task. This result thus connects or unifies CCL and OOD detection. Guided by the theoretical results, their proposed methods based on a

parameter-isolation approach and some OOD detection algorithms outperform the existing state-of-the-art baselines by a large margin. Interestingly, due to the unification of CCL and OOD detection, the theory is also applicable to open world continual learning (Kim et al. 2023a).

Several recent works have also attempted to deal with knowledge transfer and catastrophic forgetting (CF) together. For example B-CL (Ke et al. 2021c) and CTR (Ke et al. 2021a) train a shared adapter for all tasks in a pre-trained language model. They prevent CF via task masks (Serra et al. 2018) (parameter isolation) and achieve knowledge transfer via capsule networks (Sabour et al. 2017). CLASSIC (Ke et al. 2021b) achieves knowledge transfer via contrastive learning. Konishi et al. (2023) proposed a mechanism called *parameter-level soft-masking* for CF prevention and knowledge transfer. CAT (Ke et al. 2020) learns a sequence of similar and dissimilar tasks and it also deals with both CF and knowledge transfer.

Since this book is mainly related to NLP, a specific survey of settings and techniques for continual learning of various NLP tasks can be found in Ke and Liu (2022), which often needs both CF presentation and knowledge transfer.

Limitations. One key limitation of the existing CL paradigm is that the tasks and their training data are given by the user. This means that the system is not autonomous and cannot learn by itself. In order to do that, we extend the CL architecture in Fig. 2.2 to enable open-world on-the-job learning to achieve full SOLA.

2.5 The SOLA Framework

The SOLA architecture is given in Fig. 2.3, which adds the orange-colored links and associated components to the continual learning architecture in Fig. 2.2. These newly added links and components enable the system to learn by itself to achieve autonomy, which is what SOLA aims to achieve (Liu et al. 2023). It is called **learning after deployment** or **learning on the job** during application or after model deployment.

Learning after deployment refers to learning after the model has been deployed in an application or during model application (Liu and Mazumder 2021) on the fly. The basic idea is that during application, if the system/agent encounters anything that is *out-of-distribution* (OOD) or novel, it needs to **detect the novelty**. Based on the novelty, the system **creates a new task** to learn and also acquires the **ground-truth training data** to learn the task on the initiation of the system itself through interactions with humans and the environment. The system then **learns the new task incrementally** or continually. The whole process is carried out on the fly during application.

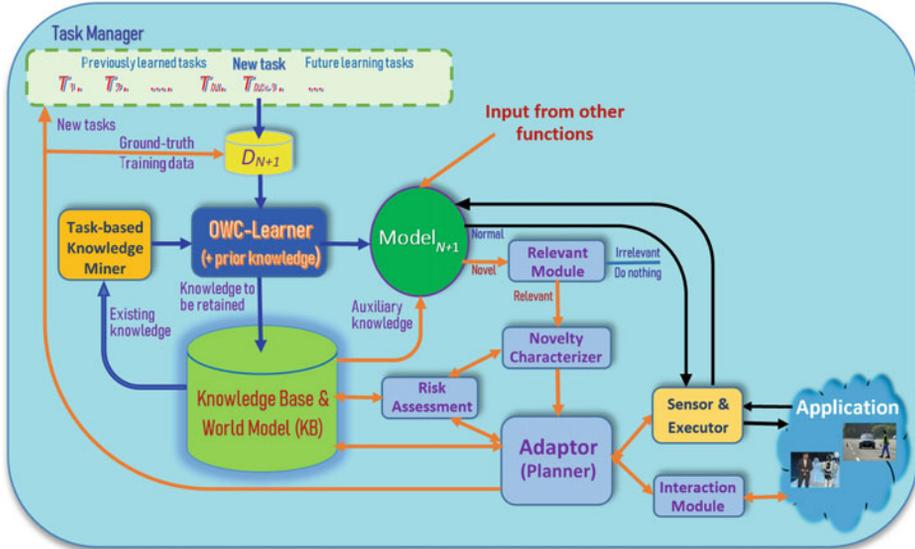


Fig. 2.3 SOLA architecture of the primary task performer or any supporting function. OWC-Learner means *Open-World Continual Learner*

2.5.1 Components of SOLA

SOLA is proposed as a framework for autonomous AI agents. An AI agent consists of a pair of key modules (P, S), where P is the *primary task-performer* that performs its main or performance task (e.g., the dialogue system of the greeting bot) and S is a *set of supporting or peripheral functions* (e.g., the vision system and the speech system of the bot) that supports the primary task-performer. The primary task-performer P or each supporting function $S_i \in S$ consist of eight core sub-systems (L, M, K, R, C, A, U, I). Figure 2.3 shows the relationships and functions of the sub-systems. We do not distinguish P and S_i in terms of techniques or subsystems as we believe they have no fundamental differences.

- **L** is the **OWC-Learner** (*Open-World Continual Learner*) that builds models to not only classify the input into known classes but also detect novel objects that have not been seen in training. For example, for the greeting bot, L of the primary task performer P is a continual learning dialogue system similar to that in Sect. 2.6. For the supporting vision system, L is a continual learner that can learn to recognize guests and detect novel or unknown objects. Compared to C-Learner in continual learning in Fig. 2.2, OWC-Learner in SOLA not only can learn continually like C-Learner but also produce models that detect novel instances in testing or in application deployment.
- **K** is the **Knowledge Base & World Model (KB)** that is important for the performance task, supporting functions or the OWC-Learner. KB is also in Fig. 2.2 but plays more

roles in SOLA. Apart from keeping the learned or prior knowledge of the domain and the world model, if needed, reasoning capability may also be provided to help the other modules of the system (see the orange-colored links). Some knowledge from the application observed by the Adapter (see below) may be added to the KB, which can provide additional knowledge to the Model for its decision making. The world model refers to the representation of the task environment and the commonsense knowledge about the objects and their relationships within.

- **M** is the **Model** learned by *L*. *M* takes the input or perception signals from the application environment to make a decision to perform actions in the application. *M* may also use some input or knowledge from other supporting functions.
- **R** is the **Relevance Module** or focusing mechanism that decides whether the detected novelty is relevant to the current task or not. If it's relevant, the agent should respond to the novelty (discussed below); otherwise ignore it. For example, in the greeting bot application, when the bot hears something from people who are chatting with each other, whether understandable or not, it will ignore them as they are irrelevant to its performance task.
- **C** is the **Novelty Characterizer** that characterizes the detected novelty based on the knowledge in the KB so that the adaptor (below) can formulate a course of actions to respond or adapt to the novelty. For the characterizer *C* of *P* of the greeting bot, as *P* is a dialogue system, when it cannot understand the utterance of a hotel guest (a novelty), it should decide what it can and cannot understand (see Sect. 2.6) and ask the guest based on its partial understanding (see below). In the case of the supporting vision system, when a novel object it detected, the characterizer may decide what the object looks like and its physical attributes. For example, the novel object may look like a dog based on the greeting bot's KB (see Sect. 2.5.4 for more discussions).
- **A** is the **Adaptor** that adapts to or accommodates the novelty based on the characterization result. It is a *planner* that produces a plan of actions for the executor *E* or the interactive module *I* to perform. Given the characterization (e.g., partial understanding) above, *A* may adapt by asking the guest to clarify (see Sect. 2.6) and then learn to understand the utterance. In the case of the vision system, if the characterizer believes that the novel object looks like a dog, the adaptor may decide to report to a hotel employee and then learns the new object by taking some pictures as the training data and asking the hotel employee for the name of the object as the class label. In the latter two cases, *A* needs to invoke *I* to interact with the human and *L* to learn the novelty so that it will not be novel in the future. That is, *A* is also responsible for creating new tasks (e.g., learning to recognize new objects by the greeting bot) on the fly and proceeds to acquire ground truth training data with the help of *I* (discussed below) to be learned by *L*. This adaptation process often involves reasoning.
- **U** is the **Risk Assessment** module. Novelty implies uncertainty in adapting to the novel situation. In making each response decision, risk needs to be assessed (see Sect. 2.5.5 for more discussions).

- **I** is the **Interactive Module** for the agent to communicate with humans or other agents, e.g., to acquire ground-truth training data or to get instructions when the agent does not know what to do in a unfamiliar situation. It may use the natural language (for interaction with humans) or an agent language (for interaction with other agents).

Several remarks are in order. First, not all agents need all these sub-systems and some sub-systems may also be shared. For example, the primary task performer P in the greeting bot application is the dialogue system. Its interaction module I can use the same dialogue system. In some cases, the Model may also be able to determine the relevance of a novel object to the application and even characterize the novelty because characterization in many cases is about classification and similarity comparison. Second, as we will see, every sub-system can and should have its own local learning capability. Third, the interaction module I and the adapter A will create new tasks to learn and gather ground truth training data for learning. Fourth, most links in Fig. 2.3 are bidirectional, which means that the sub-systems need to interact with each other in order to perform their tasks. The interactions may involve requesting for information, passing information, and/or going back and forth with hypothesis generation and hypothesis evaluation to make more informed decisions.

Since the primary task performer P and each supporting sub-system S_i has the same components or sub-systems, we will discuss them in general rather than distinguishing them. Below, we first discuss the SOLA learner L .

2.5.2 Open World Continual Learning

The core learning capability of SOLA is *open world continual learning*. However, as discussed earlier, the classical ML paradigm makes the *closed world assumption*, which is often violated in practice. Here we first define some concepts and then present the paradigm of *open-world continual learning* (OWC-Learning).

Let the training data that have been seen so far from previous tasks be $\mathcal{D}_{tr} = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}_{tr}$. Let the set of class labels that may appear in testing or application be \mathbf{Y}_{tst} . Classical ML makes the closed-world assumption.

Closed-world assumption: There are no new or novel instances or classes that may appear in testing or application, i.e., $\mathbf{Y}_{tst} \subseteq \mathbf{Y}_{tr}$. In other words, every class seen in testing or application must have been seen in training.

Open world: There are test classes that have not been seen in training, i.e., $\mathbf{Y}_{tst} - \mathbf{Y}_{tr} \neq \emptyset$.

Definition (Closed-world learning): It refers to the learning paradigm that makes the closed-world assumption.

Definition (Open-world learning (OWL)): It refers to the learning paradigm that performs the following functions: (1) classify test instances belonging to training classes to their respective classes and detect novel or out-of-distribution (OOD) instances, and

(2) learn the novel classes labeled by humans in the identified novel instances to update the model using the labeled data. The model updating is initiated by human engineers and involves re-training or incremental learning. If incremental learning is used, it is the class continual learning (CCL).

Definition (Open-world continual learning (OWC-learning)): OWC-learning is the learning paradigm that performs open-world learning but the learning process is initiated by the agent itself after deployment with no involvement of human engineers. The new task creation and ground-truth training data acquisition are done by the agent via its interaction with the user and the environment. The learning of the new task is incremental, i.e., CCL with no re-training of previous tasks/classes. The process is lifelong or continual.

The core learning capabilities of OWL and OWC-learning are OOD detection and CCL. Kim et al. (2023a) shows that the theoretical result in Kim et al. (2022, 2023b) also forms the theory for the learning aspects of the open-world continual learning.

2.5.2.1 Steps in OWC-Learning

The main process in OWC-Learning involves the following three steps, which can be regarded as part of the novelty adaptation or accommodation strategy (see Sect. 2.5.4).

Step 1—Novelty detection. This step involves detecting data instances whose classes do not belong to \mathbf{Y}_{tr} . As mentioned earlier, a fair amount of research has been done on this under open-set classification or out-of-distribution (OOD) detection (Pang et al. 2021).

Step 2—Acquiring class labels and creating a new learning task on the fly: This step first clusters the detected novel instances. Each cluster represents a new class. Clustering may be done automatically or through interaction with humans using the interaction module I . Interacting with human users should produce more accurate clusters and also obtain meaningful class labels. If the detected data is insufficient to learn an accurate model to recognize the new classes, additional ground-truth data may be collected via interaction with human users (and/or passively by downloading data from the Web, like searching and scrapping images of objects of a given class). A new learning task is created.

In the case of our hotel greeting bot, since the bot detects a single new guest (automatically), no clustering is needed. It then asks the guest for his/her name as the class label. It also takes more pictures as the training data. With the labeled ground-truth data, a new learning task is created to incrementally learn to recognize the new guest on the fly.

The learning agent may also interact with the environment to obtain training data. In this case, the agent must have an *internal evaluation system* that can assign rewards to different states of the world, e.g., for reinforcement learning.

Step 3—Incrementally learn the new task. After ground-truth training data has been acquired, the OWC-Learner L incrementally learns the new task. This is the *CCL setting of continual learning*. We will not discuss it further as there are already numerous existing techniques (see Sect. 2.4).

2.5.2.2 Acquiring Training Data Automatically

Existing approaches to obtaining the training data is through manual labeling or writing (in the case of dialogue data), which is both costly and time-consuming. In the case of chatbots, as they typically work in multi-user environments, we can exploit such environments to acquire the ground truth training data interactively during actual online conversations. This process is both automatic and free. We have discussed this issue in Sect. 1.4 and we briefly reproduce it here for completeness of this section.

1. Extracting data or information directly from user utterances (or dialogue history), which can be real-world facts, user preferences, etc. Additional knowledge/data may be inferred from the acquired and existing KB knowledge. Related works include Kollar et al. (2013) and Padmakumar et al. (2017).

2. Asking the current user when the agent (1) doesn't understand a user utterance, or (2) cannot answer a user query, which forms a new learning task. To obtain the ground truth data, for (1), the agent may ask the current user for clarification or rephrasing. For (2), it may ask the user for some supporting facts and then infer the query answer (Mazumder et al. 2019, 2020a; Padmakumar and Mooney 2021). In this process, it obtains command-action pairs, question-answer pairs, etc.

3. Asking other users to obtain the answers when the chatbot could not answer a user query. For example, if a user asks "*What is the capital city of the US?*" and the chatbot is unable to answer or infer now, it can try to find a good opportunity to ask another user later "*Hey, do you happen to know what the capital city of the US is?*" If the user gives the answer "*it's Washington DC,*" the agent acquires the ground truth (a piece of new knowledge) which can be used in its future conversations or as a piece of training data. Some related approaches can be found in the works on Opportunistic Active Learning (Padmakumar et al. 2018).

4. Observing user demonstrations if supported. See Sect. 1.4 and also Mazumder et al. (2020b).

Beyond these, the agent may also extract ground-truth data from online documents or online knowledge bases.

2.5.3 Relevance of Novelty

As an AI agent has a performance task, it should focus on novelties that are critical to the performance task. For example, a self-driving car should focus on novel objects or events that are or may potentially appear on the road in front of the car. It should not pay attention to novel objects in the shops along the street (off the road) as they do not affect driving. This relevance check involves gathering information about the novel object to make a classification decision. As this is a normal classification task, it is not discussed further.

2.5.4 Novelty Characterization and Adaptation

In a real-life application, classification may not be the primary task of the agent. For example, in a self-driving car, object classification or recognition supports its primary performance task of driving. To drive safely, the car has to take some actions to adapt or respond to any novel/new objects, e.g., slowing down and avoiding the objects. In order to know what actions to take to adapt, the agent must characterize the new object. The **characterization** of a novel object is a partial description of the object based on the agent's existing knowledge about the world. According to the characterization, appropriate actions are formulated to **adapt** or respond to the novel object. The process may also involve learning.

Novelty characterization and *adaptation* (or response) form a pair (c, r) , where c is the *characterization of the novelty* and r is the *adaptation response* to the novelty, which is a plan of dynamically formulated actions based on the characterization of the novelty. The two activities go hand-in-hand. Without an adaptation strategy for a characterization, the characterization has little use. If the system cannot characterize a novelty, it takes a low risk-assessed *default response*. In our greeting bot example, when it can characterize a novelty as a new guest, its response is to say “*Hello, welcome to our hotel! What is your name, sir?*” If the bot has difficulty with characterization, it can take a *default action*, e.g., ‘do nothing.’ The set of responses are specific to the application. For a self-driving car, the default response to a novel object is to slow down or stop the car so that it will not hit the object. In some situations, the agent must take an action under low confidence circumstances, e.g., the agents engage in reinforcement learning, trying actions and assessing outcomes.

Characterization can be done at different levels of detail, which may result in more or less precise responses. Based on an ontology and object attributes related to the performance task in the domain, characterization can be described based on the *type of the object* and the *attributes of the object*. For example, in the greeting bot application, it is useful to determine whether the novel object is a human or an animal because the responses to them are different. For self-driving cars, when sensing a novel object on the road, it should focus on those aspects that are important to driving, e.g., whether it is a still or a moving object. If it is a moving object, the car should determine its direction and speed of moving. Thus, classification of movement is needed in this case to characterize the novelty, which, in turn, facilitates determination of the agent's responding action(s). For instance, if the novel object is a mobile object, the car may wait for the object to leave the road before driving.

Another characterization strategy is to *compare the similarity* between the novel object and the existing known objects. For example, if it is believed that the novel object looks like a dog (assuming the agent can recognize a dog), the agent may react like when it sees a dog on the road.

The above discussion implies that in order to effectively characterize a novelty, the agent must already have a great deal of world knowledge that it can use to describe the novelty. Additionally, the characterization and response process is often interactive in the sense that the agent may choose a course of actions based on the initial characterization. After some

actions are taken, it may get some feedback from the environment. Based on the feedback and the agent’s additional observations, the course of actions may change.

Learning to respond. In some situations, the system may not know how to respond to a novel object or situation. It may try one or more of the following ways.

(1) *Asking a human user* (Mosqueira-Rey et al. 2022; Ren et al. 2021; Settles 2009; Wu et al. 2022). In the case of the self-driving car, when it does not know what to do, it may ask the passenger using the interactive module *I* in natural language and then follow the instruction from the passenger and also learn it for future use. For example, if the car sees a black patch on the road that it has never seen before, it can ask “*what is that black thing in front?*” The passenger may answer “*that is tar.*” If there is no ready response, e.g., no prior information about tar, the system may progress with a further inquiry, asking the passenger “*what should I do?*”

(2) *Imitation learning* (Fang et al. 2019; Hussein et al. 2017; Torabi et al. 2019; Zheng et al. 2022). On seeing a novel object, if the car in front drives through it with no issue, the car may choose the same course of action as well and also learn it for future use if the car drives through without any problem.

(3) *Reinforcement learning* (Arzate Cruz and Igarashi 2020; Li 2017; Sutton and Barto 2018). By interacting with the environment through trial and error exploration, the agent learns a good response policy. This can be extremely challenging and risky as any action taken has consequences and cannot be reversed. For this to work, the agent must have an internal evaluation system that is able to assign rewards to (possible) states.

If multiple novelties are detected at the same time, it is more difficult to respond as the agent must reason over the characteristics of all novel objects to dynamically formulate an overall plan of actions that prioritize the responses.

2.5.5 Risk Assessment and Learning

There is risk in achieving performance goals of an agent when making an incorrect decision. For example, classifying a known guest as unknown or an unknown guest as known may negatively affect guest impressions resulting in negative reviews. For a self-driving car, misidentifications can result in wrong responses, which could be a matter of life and death. Thus, risk assessment must be done in making each decision. Risk assessment can also be learned from experiences or mistakes. In the example of a car passing over tar, after the experience of passing over shiny black surfaces safely many times, if the car slips in one instance, the car agent must assess the risk of continuing the prior procedure. Given the danger, a car may weight the risk excessively, slowing down on new encounters of shiny black surfaces.

Safety net: Another aspect of risk is safety net. If a wrong decision is made, it should not cause a catastrophe. A wrong decision may be due to two reasons. The first is the model inaccuracy. For example, the perception system of a self-driving car may classify a

pedestrian wearing a black dress as a patch of tar on the surface of the road, and run over the person. The second is wrong knowledge acquired from human users during human-agent interaction. For example, if a self-driving car sees an unknown object (e.g., a big stone) in the middle of the road and asks the passenger what it is and what to do, the passenger may say it is a stone and it is safe to drive through it. In this case, the car should not go ahead and hit the stone.

2.6 A Dialogue System based on SOLA

We now briefly describe a dialogue system (called CML) in Mazumder et al. (2020b) that uses the SOLA framework and learns continually by itself on the fly during conversation with users to become more and more powerful. Details will appear in Sect. 4.2. Another two related systems (Mazumder et al. 2019; Mazumder et al. 2020a) can be found in Sect. 3.4, which are more sophisticated than CML.

CML is a natural language interface system like Amazon Alexa and Apple Siri. Its **performance task** is to take a user command in natural language (NL) and perform the user requested API action in the underlying application, e.g., “*switch off the light in the bedroom.*” Since CML is a text-based research system, it does not have any other support functions or sub-systems. The key issue of CML is to understand paraphrased NL commands from the user in order to map a user command to a system’s API call. The **Model** of CML consists of two parts. The first part is a store S of *seed commands* (SCs). Each SC in S is similar to an actual user command except that the arguments used by the associated API function is replaced by variables, e.g., “*switch off the light in the X,*” where X is the variable representing the argument of the API function for this command. A SC is either prepared by the API developer before the system is deployed or continually learned from users on the fly during conversation. The second part is a command matching system that matches a user command to a saved SC in S . If no match can be found, it means an unexpected or *novel command* from the user is detected, i.e., a *novelty*, which equates to the system’s failure in understanding or grounding a user command. The novelty detection task is performed by the **Model**. CML assumes that every novelty is **relevant** to the performance task, but if needed, a **Relevance Module** can be built as a classifier to classify whether a user command is relevant or not to the application. **Novelty characterization** of CML is also performed by the **Model**. In the matching process, CML tries to identify the part of the user command that the system does not understand and how similar it is to some known SCs in S . Based on the characterization, the **Adaptor** dynamically formulates an interaction strategy based on the context (e.g., dialogue history, command from the user, information acquired from the user, outstanding information needed to complete a task, etc.) and executes the strategy to carry out multi-turn dialogues for knowledge acquisition (i.e., obtaining the ground truth data from the user). Specifically, in CML, it decides *when to ask* and *what to ask* the user, to achieve two goals: (1) understand the user command so that it can perform the correct

API action for the user, and (2) create a new task and use the correct API action as the ground-truth for the user command for learning by **OWC-Learner**. CML uses a rule-based system for adaptation. Note that a finite state machine is used in another system (Mazumder et al. 2020a). Based on the user’s command and the ground-truth API action, **OWC-Learner** learns this command incrementally by creating a new SC and adding it to the SC store S so that the system will be able to match this and similar commands in the future. In the adaptation or accommodation process, **risk** is also considered (see below). There is no separate **Interaction Module** because the main task performer is a dialogue system, which is an interactive system itself and can serve the role of the Interaction Module.

Consider the following example user command “*turn off the light in the kitchen*” that the **Model** cannot match or ground to an existing SC (i.e., it does not understand). A **novelty** is reported. The **Model** also **characterizes** the novelty by deciding which part of the command it can match or understand, which part it has difficulty with, and what known SCs are similar to the user command. Based on the characterization result, the **Adaptor** formulates actions to be taken. In this case, it provides the user a list of *top-k* predicted actions (see below) described in natural language and asks the user to select the most appropriate action from the given list.

Bot: Sorry, I didn’t get you. Do you mean to:

- option-1.** switch off the light in the kitchen, or
 - option-2.** switch on the light in the kitchen, or
 - option-3.** change the color of the light?
-

The user selects the desired action (option-1). The action API [say, `SwitchOffLight(arg:place)`] corresponding to the selected action (option-1) is retained as the ground truth action for the user-issued command. The **Adaptor** invokes the **Executor** to perform the API action for the user. In subsequent turns of the dialogue, the system can also ask the user questions to acquire ground truth values associated with arguments of the selected action, as defined in the API. Based on the acquired information, **OWC-Learner** of CML **incrementally learns** to map the original command “*turn off the light in the kitchen*” to the API action, `SwitchOffLight(arg:place)`, by creating a new SC (“*turn off the light in the X*”) and adding it to the SC store, which ensures that in the future the system will not have problem understanding the related commands.

Risk is considered in CML in two ways. First, it does not ask the user too many questions in order not to annoy the user. Second, when the characterization is not confident, the **Adaptor** simply asks the user to say his/her command again in an alternative way, i.e., “*Can you say it again in another way?*” (the new way may be easier for the system to understand) rather than providing a list of random options for the user to choose from. If the options have nothing to do with the user command, the user may lose confidence in the system.

2.7 Comparison with Related Learning Paradigm

This section summarizes the difference between the SOLA framework and some existing machine learning frameworks or paradigms. There are three machine learning paradigms that are closely related to SOLA, i.e., *novelty detection*, *open-world learning*, and *continual learning*. We compare SOLA with these three topics below.

2.7.1 Novelty Detection

Novelty detection is also known as *anomaly detection*, *out-of-distribution (OOD) detection*, *open set detection*, or *open classification*. Extensive research has been done on novelty detection (Fei and Liu 2016; Xu et al. 2019). Two recent surveys of the topic can be found in Pang et al. (2021) and Yang et al. (2021a). Novelty detection is only the first step in SOLA and it does not involve lifelong or continual learning, which is the backbone of SOLA, and it also does not involve novelty characterization and adaptation or the associated risk assessment. Some of the related works on novelty or OOD detection in the context of dialogue systems are (Kim et al. 2020b; Lin and Xu 2019a; Lin et al. 2020; Shu et al. 2021; Yan et al. 2020; Zhang et al. 2021b).

2.7.2 Open-World Learning

Some researchers have also studied learning the novel objects after they are detected and manually labeled (Bendale and Boult 2015; Fei et al. 2016; Xu et al. 2019), which are called *open-world learning*. A survey of the topic is given in Parmar et al. (2023). A position paper by Langley (2020) presented some blue sky ideas about open-world learning, but it does not have sufficient details or an implemented system. SOLA differs from open-world learning in many ways. The key difference is that in the current work on open-world learning, the tasks and their training data are given by the user or engineers, while SOLA stresses self-initiation in learning, which means that all the learning activities from start to end are *self-motivated* and *self-initiated* by the agent itself. The process involves no human engineers, which is important for AI autonomy. Furthermore, the research in open-world learning so far has not involved continual learning of a sequence of tasks and thus has not dealt with the issue of catastrophic forgetting and knowledge transfer.

Due to self-initiation, SOLA enables learning after the model deployment like human learning on the job or while working, which has barely been attempted before. In existing learning paradigms, after a model has been deployed, there is no more learning until the model is updated or retrained on the initiation of human engineers. However, there are works (Kollar et al. 2013; Mazumder et al. 2020a, b; Padmakumar et al. 2018; She and Chai 2017) that have made some initial efforts along this direction.

2.7.3 Continual Learning

Continual learning (CL) aims to learn a sequence of tasks incrementally. However, existing research assumes that the tasks and their training data are given by users or engineers. SOLA differs from CL in two main ways.

(1) SOLA includes modules to characterize and adapt to novel situations so that the agent can work in the open world environment without stopping. These are not included in continual learning.

(2) SOLA makes learning autonomous and self-initiated. This involves online interactions of the learning agent with human users, other AI agents, and the environment. The purpose is to acquire ground-truth training data on the fly by the agent itself. This is very similar to what we humans do when we encounter something novel or new and ask others interactively and learn it. It is very different from collecting a large amount of unlabeled data and asking human annotators to label the data. Incidentally, SOLA also differs from active learning (Ren et al. 2021; Settles 2009) as active learning only focuses on acquiring labels from users for selected unlabeled examples in a given dataset.

Finally, note that although SOLA focuses on self-initiated continual learning, it does not mean that the learning system cannot learn a task given by humans or other AI agents. Additionally, SOLA also allows learning from other resources, e.g., the Web, to gain new knowledge, like a human reading a book. We will describe the system NELL (Mitchell et al. 2015) that continuously learns new knowledge by extracting facts from Web documents in Sect. 3.5.

2.8 Summary

This chapter presented a new framework called *self-initiated open world continual learning and adaptation* (SOLA) that should be followed by AI agents (including chatbots or dialogue systems) that want to continually learn after model deployment or learn on the job. We believe that this capability is necessary for the next generation machine learning or AI agents and any form of intelligence in general. The core of this framework is self-motivation and self-initiation. That is, the AI agent must learn autonomously and continually in the open world on its own initiative after deployment (or post-deployment), detecting novelties, adapting to the novelties and the ever-changing world, and learning more and more to become more and more powerful over time. An example SOLA based dialogue system (called CML) is briefly described (see Sect. 4.2 for more details of the system). Almost all existing dialogue systems related to lifelong learning are only partial continual learning dialogue systems. They do not include all functions in the proposed framework. For example, some only perform continual learning with given tasks and training data, some only detect novelties, and some only continually collect additional training data during chatting.

References

- H. Ahn, S. Cha, D. Lee, T. Moon, Uncertainty-based continual learning with adaptive regularization, in *Advances In Neural Information Processing Systems*, vol. 32 (2019)
- C. Arzate Cruz, T. Igarashi, A survey on interactive reinforcement learning: Design principles and open challenges, in *Proceedings of the 2020 ACM Designing Interactive Systems Conference* (2020), pp. 1195–1209
- A. Bendale, T. Boulton, Towards open world recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1893–1902
- T.E. Boulton, P.A. Grabowicz, D.S. Prijatelj, R. Stern, L. Holder, J. Alspector, M. Jafarzadeh, T. Ahmad, A.R. Dhamija, C. Li et al., Towards a unifying framework for formal theories of novelty, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35 (2021), pp. 15047–15052
- P. Buzzega, M. Boschini, A. Porrello, D. Abati, S. Calderara, Dark experience for general continual learning: a strong, simple baseline. *Adv. Neural Inf. Process. Syst.* **33**, 15920–15930 (2020)
- R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, G. Metta, Incremental robot learning of new objects with fixed update time, in *IEEE International Conference on Robotics and Automation (ICRA)*. (IEEE, 2017), pp. 3207–3214
- H. Cha, J. Lee, J. Shin, Co2l: contrastive continual learning, in *Proceedings of the IEEE/CVF International conference on computer vision*, pp. 9516–9525 (2021)
- A. Chaudhry, M. Ranzato, M. Rohrbach, M. Elhoseiny, Efficient lifelong learning with a-gem, in *International Conference on Learning Representations* (2019)
- A. Chaudhry, N. Khan, P. Dokania, P. Torr, Continual learning in low-rank orthogonal subspaces. *Adv. Neural Inf. Process. Syst.* **33**, 9900–9911 (2020)
- Z. Chen, B. Liu, *Lifelong Machine learning*. (Morgan and Claypool Publishers, 2018)
- Z. Chen, B. Liu, Topic modeling using topics from many domains, lifelong learning and big data, in *International Conference on Machine Learning* (2014), pp. 703–711
- P. Dhar, R. Vikram Singh, K-C. Peng, Z. Wu, R. Chellappa, Learning without memorizing, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5138–5146
- B. Fang, S. Jia, D. Guo, X. Muhua, S. Wen, F. Sun, Survey of imitation learning for robotic manipulation. *Int. J. Intell. Robot. Appl.* **3**, 362–369 (2019)
- G. Fei, B. Liu, Breaking the closed world assumption in text classification, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2016), pp. 506–514
- G. Fei, S. Wang, B. Liu, Learning cumulatively to become more knowledgeable, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. (ACM, 2016), pp. 1565–1574
- A. Gepperth, C. Karaoguz, A bio-inspired incremental learning architecture for applied perceptual problems. *Cognit. Comput.* **8**(5), 924–934 (2016)
- Y. Guo, B. Liu, D. Zhao, Online continual learning through mutual information maximization, in *International Conference on Machine Learning*. (PMLR, 2022b), pp. 8109–8126
- Y. Guo, H. Wenpeng, D. Zhao, B. Liu, Adaptive orthogonal projection for batch and online continual learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36 (2022), pp. 6783–6791
- Haitham Bou Ammar, Eric Eaton, José Marcio Luna, and Paul Ruvolo (Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning, In Twenty-fourth international joint conference on artificial intelligence, 2015)
- G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network (2015). [arXiv:1503.02531](https://arxiv.org/abs/1503.02531)

- A. Hussein, M.M. Gaber, E. Elyan, C. Jayne, Imitation learning: a survey of learning methods. *ACM Comput. Surv.(CSUR)*, **50**(2), 1–35 (2017)
- H. Jung, J. Ju, M. Jung, J. Kim, Less-forgetting learning in deep neural networks (2016). [arXiv:1607.00122](https://arxiv.org/abs/1607.00122)
- N. Kamra, U. Gupta, Y. Liu, Deep generative dual memory network for continual learning (2017). [arXiv:1710.10368](https://arxiv.org/abs/1710.10368)
- Z. Ke, B. Liu, Continual learning of natural language processing tasks: A survey (2022). [arXiv:2211.12701](https://arxiv.org/abs/2211.12701)
- Z. Ke, B. Liu, H. Xu, N. Ma, L. Shu, Achieving forgetting prevention and knowledge transfer in continual learning. *Adv. Neural Inf. Process. Syst.* **34**, 22443–22456 (2021a)
- Z. Ke, B. Liu, H. Xu, L. Shu, classic: continual and contrastive learning of aspect sentiment classification tasks, in *Proceedings of 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP-2021)* (2021b)
- Z. Ke, H. Xu, B. Liu, Adapting bert for continual learning of a sequence of aspect sentiment classification tasks, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2021c), pp. 4746–4755
- Z. Ke, B. Liu, X. Huang, Continual learning of a mixed sequence of similar and dissimilar tasks. *Adv. Neural Inf. Process. Syst.* **33**, 18493–18504 (2020)
- R. Kemker, C. Kanan, FearNet: brain-inspired model for incremental learning, in *International Conference on Learning Representations* (2018)
- S. Kim, M. Eric, K. Gopalakrishnan, B. Hedayatnia, Y. Liu, D. Hakkani-Tur, Beyond domain apis: task-oriented conversational modeling with unstructured knowledge access, in *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2020b), pp. 278–289
- G. Kim, S. Esmailpour, C. Xiao, B. Liu, Continual learning based on ood detection and task masking, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022a), pp. 3856–3866
- G. Kim, C. Xiao, T. Konishi, Z. Ke, B. Liu, A theoretical study on solving continual learning. **35**, 5065–5079 (2022)
- G. Kim, C. Xiao, T. Konishi, Z. Ke, B. Liu, Open-world continual learning: Unifying novelty detection and continual learning (2023a). [arXiv:2304.10038](https://arxiv.org/abs/2304.10038) [cs.LG]
- G. Kim, C. Xiao, T. Konishi, B. Liu, Learnability and algorithm for continual learning, in *Proceedings of Fortieth International Conference on Machine Learning (ICML-2023)* (2023b)
- J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, Others, Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci.* **114**(13), 3521–3526 (2017)
- T. Kollar, V. Perera, D. Nardi, M. Veloso, Learning environmental knowledge from task-based human-robot dialog, in *2013 IEEE International Conference on Robotics and Automation*. (IEEE, 2013), pp. 4304–4309
- T. Konishi, M. Kurokawa, C. Ono, Z. Ke, G. Kim, B. Liu, Parameter-level soft-masking for continual learning, in *Proceedings of Fortieth International Conference on Machine Learning (ICML-2023)* (2023)
- P. Langley, Open-world learning for radically autonomous agents. *Proc. AAAI Conf. Artif. Intell.* **34**, 13539–13543 (2020)
- K. Lee, K. Lee, J. Shin, H. Lee, Overcoming catastrophic forgetting with unlabeled data in the wild, in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 312–321
- Y. Li, Deep reinforcement learning: An overview (2017). [arXiv:1701.07274](https://arxiv.org/abs/1701.07274)
- Z. Li, D. Hoiem, Learning without forgetting, in *European Conference on Computer Vision*. (Springer, 2016), pp. 614–629

- T-E. Lin, X. Hua, H. Zhang, Discovering new intents via constrained deep adaptive clustering with cluster refinement. *Proc. AAAI Conf. Artif. Intell.* **34**, 8360–8367 (2020)
- T-E. Lin, H. Xu, Deep unknown intent detection with margin loss, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019a), pp. 5491–5496
- B. Liu, S. Mazumder, E. Robertson, S. Grigsby, *Self-initiated open-world continual learning and adaptation*. *AI Magazine*, AI autonomy (2023)
- Y. Liu, S. Parisot, G. Slabaugh, X. Jia, A. Leonardis, T. Tuytelaars, More classifiers, less forgetting: a generic multi-classifier paradigm for incremental learning, in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*. (Springer, 2020c), pp. 699–716
- Y. Liu, B. Schiele, Q. Sun, Adaptive aggregation networks for class-incremental learning, in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition* (2021b), pp. 2544–2553
- B. Liu, Learning on the job: online lifelong and continual learning. *Proc. AAAI Conf. Artif. Intell.* **34**, 13544–13549 (2020)
- B. Liu, S. Mazumder, Lifelong and continual learning dialogue systems: learning during conversation. *Proc. AAAI Conf. Artif. Intell.* **35**, 15058–15063 (2021)
- D. Lopez-Paz, M. Ranzato, Gradient Episodic Memory for Continual Learning, in *Advances in Neural Information Processing Systems* (2017), pp. 6470–6479
- N. Ma, S. Mazumder, A. Politowicz, B. Liu, E. Robertson, S. Grigsby, Semantic novelty detection and characterization in factual text involving named entities, in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. (Association for Computational Linguistics, 2022), pp. 9225–9252
- N. Ma, A. Politowicz, S. Mazumder, J. Chen, B. Liu, E. Robertson, S. Grigsby, Semantic novelty detection in natural language descriptions, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), pp. 866–882
- A. Mallya S. Lazebnik, Packnet: adding multiple tasks to a single network by iterative pruning, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2018), pp. 7765–7773
- S. Mazumder, *On-the-job Continual and Interactive Learning of Factual Knowledge and Language Grounding*. Ph.D. thesis, University of Illinois at Chicago (2021)
- S. Mazumder, B. Liu, N. Ma, S. Wang, Continuous and interactive factual knowledge learning in verification dialogues, in *NeurIPS-2020 Workshop on Human And Machine in-the-Loop Evaluation and Learning Strategies* (2020a)
- S. Mazumder, B. Liu, S. Wang, S. Esmaeilpour, An application-independent approach to building task-oriented chatbots with interactive continual learning, in *NeurIPS-2020 Workshop on Human in the Loop Dialogue Systems* (2020b)
- S. Mazumder, B. Liu, S. Wang, N. Ma, Lifelong and interactive learning of factual knowledge in dialogues, in *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue* (2019), pp. 21–31
- M. McCloskey, N.J. Cohen, Catastrophic interference in connectionist networks: The sequential learning problem, in *Psychology of learning and motivation*, vol. 24. (Elsevier, 1989), pp. 109–165
- T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, Never-ending learning, in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-15)* (2015)
- J.G. Moreno-Torres, T. Raeder, R. Alaiz-Rodríguez, N.V. Chawla, F. Herrera, A unifying view on dataset shift in classification. *Pattern Recognit.* **45**(1), 521–530 (2012)

- E. Mosqueira-Rey, E. Hernández-Pereira, D. Alonso-Ríos, J. Bobes-Bascarán, Á. Fernández-Leal, Human-in-the-loop machine learning: a state of the art. *Artif. Intell. Rev.* (2022), pp. 1–50
- O. Ostapenko, M. Puscas, T. Klein, P. Jahnichen, M. Nabi, Learning to remember: a synaptic plasticity driven framework for continual learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 11321–11329
- A. Padmakumar, R.J. Mooney, Dialog policy learning for joint clarification and active learning queries, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35 (2021), pp. 13604–13612
- A. Padmakumar, P. Stone, R. Mooney, Learning a policy for opportunistic active learning, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), pp. 1347–1357
- A. Padmakumar, J. Thomason, R. Mooney, Integrated learning of dialog strategies and semantic parsing, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers* (2017), pp. 547–557
- G. Pang, C. Shen, L. Cao, A. Van Den Hengel, Deep learning for anomaly detection: a review. *ACM Comput. Surv. (CSUR)* **54**(2), 1–38 (2021)
- J. Parmar, S. Chouhan, V. Raychoudhury, S. Rathore, Open-world machine learning: applications, challenges, and opportunities. *ACM Comput. Surv.* **55**(10), 1–37 (2023)
- Q. Qin, H. Wenpeng, H. Peng, D. Zhao, B. Liu, Building network structures dynamically for continual learning, *Bns. Adv. Neural Inf. Process. Syst.* **34**, 20608–20620 (2021)
- J. Rajasegaran, M. Hayat, S. Khan, F. Shahbaz Khan, L. Shao, M-H. Yang, An adaptive random path selection approach for incremental learning (2019). [arXiv:1906.01120](https://arxiv.org/abs/1906.01120)
- S-A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, icarl: incremental classifier and representation learning, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2017), pp. 2001–2010
- P. Ren, Y. Xiao, X. Chang, P-Y. Huang, Z. Li, B.B. Gupta, X. Chen, X. Wang, A survey of deep active learning. *ACM Comput. Surv. (CSUR)* **54**(9), 1–40 (2021)
- D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, G. Wayne, Experience replay for continual learning, in *Advances in Neural Information Processing Systems*, vol. 32 (2019)
- A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks (2016). [arXiv:1606.04671](https://arxiv.org/abs/1606.04671)
- P. Ruvolo, E. Eaton, Ella: an efficient lifelong learning algorithm, in *International Conference on Machine Learning* (2013), pp. 507–515
- S. Sabour, N. Frosst, G.E. Hinton, Dynamic routing between capsules, in *Advances in Neural Information Processing Systems*, vol. 30 (2017)
- J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. Whye Teh, R. Pascanu, R. Hadsell, Progress & compress: A scalable framework for continual learning, in *International Conference On Machine Learning*. (PMLR, 2018), pp. 4528–4537
- A. Seff, A. Beatson, D. Suo, H. Liu, Continual learning in generative adversarial nets (2017). [arXiv:1705.08395](https://arxiv.org/abs/1705.08395)
- J. Serra, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in *International Conference on Machine Learning*. (PMLR, 2018), pp. 4548–4557
- B. Settles, Active learning literature survey (2009)
- L. She, J. Chai, Interactive learning of grounded verb semantics towards human-robot communication, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017), pp. 1634–1644
- H. Shin, J.K. Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay, in *Advances in Neural Information Processing Systems*, vol. 30 (2017)
- L. Shu, Y. Benajiba, S. Mansour, Y. Zhang, Odism, Open world classification via distributionally shifted instances. *Find. Assoc. Comput. Linguist.: EMNLP* **2021**, 3751–3756 (2021)

- D.L. Silver, Q. Yang, L. Li, Lifelong machine learning systems: beyond learning algorithms, in *AAAI Spring Symposium: Lifelong Machine Learning*. (Citeseer, 2013), pp. 49–55
- R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction*. (MIT press, 2018)
- S. Thrun, T.M. Mitchell, Lifelong robot learning. *Robot. Autonom. Syst.* **15**(1–2), 25–46 (1995)
- F. Torabi, G. Warnell, P. Stone, Recent advances in imitation learning from observation, in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence* (2019), pp. 6325–6331
- G.M. van de Ven, A.S. Tolias, Three scenarios for continual learning (2019). [arXiv:1904.07734](https://arxiv.org/abs/1904.07734)
- J. von Oswald, C. Henning, B.F. Grewe, J. Sacramento, Continual learning with hypernetworks, in *8th International Conference on Learning Representations (ICLR 2020)*. International Conference on Learning Representations (2020)
- H. Wenpeng, Z. Lin, B. Liu, C. Tao, Z. Tao Tao, D. Zhao, J. Ma, R. Yan, Overcoming catastrophic forgetting for continual learning via model adaptation, in *International Conference on Learning Representations* (2019)
- M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, A. Farhadi, Supermasks in superposition. *Adv. Neural Inf. Process. Syst.* **33**, 15173–15184 (2020)
- W. Xingjiao, L. Xiao, Y. Sun, J. Zhang, T. Ma, L. He, A survey of human-in-the-loop for machine learning. *Futur. Gener. Comput. Syst.* **135**, 364–381 (2022)
- H. Xu, B. Liu, L. Shu, P Yu, Open-world learning and application to product classification, in *The World Wide Web Conference* (2019), pp. 3413–3419
- G. Yan, L. Fan, Q. Li, H. Liu, X. Zhang, Xi-M. Wu, A.Y.S. Lam, Unknown intent detection using gaussian mixture model with an application to zero-shot intent classification, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 1050–1060
- J.Yang, K. Zhou, Y. Li, Z. Liu, Generalized out-of-distribution detection: a survey (2021a). [arXiv:2110.11334](https://arxiv.org/abs/2110.11334)
- F. Zenke, B. Poole, S. Ganguli, Continual learning through synaptic intelligence, in *International Conference on Machine Learning* (2017), pp. 3987–3995
- H. Zhang, X. Hua, T-E. Lin, R. Lyu, Discovering new intents with deep aligned clustering. *Proc. AAAI Conf. Artif. Intell.* **35**, 14365–14373 (2021)
- B. Zheng, S. Verma, J. Zhou, I.W. Tsang, F. Chen, Imitation learning: Progress, taxonomies and challenges. *IEEE Trans. Neural Netw. Learn. Syst.* 1–16 (2022)
- F. Zhu, X-Y. Zhang, C. Wang, F. Yin, C-L. Liu, Prototype augmentation and self-supervision for incremental learning, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 5871–5880



Continuous Factual Knowledge Learning in Dialogues

3

Lifelong or continual knowledge learning in an interactive environment is a key capability of human beings. One can only learn so much by being told or supervised because the world is simply too complex to be completely learned this way. In fact, we humans probably learn a great deal of our knowledge through interactions with other humans and the environment around us which constantly give us explicit and implicit feedback. In the context of chatbots, continual and interactive knowledge learning is critical because in order for a chatbot to be truly intelligent in human-machine conversation, it has to continually learn new knowledge and improve itself over time.

Chatbots had a long history in AI and natural language processing (NLP). They became particularly and increasingly popular in the past few years due to the commercial success of some chatbots or virtual assistants such as Echo and Siri. Numerous chatbots have been developed or are under development, and many researchers are also actively working on techniques for chatbots. Early chatbot systems were mostly built using markup languages such as AIML,¹ handcrafted conversation generation rules, and/or information retrieval techniques (Ameixa et al. 2014; Banchs and Li 2012; Lowe et al. 2015; Serban et al. 2018). Recent neural conversation models (Li et al. 2017b; Vinyals and Le 2015; Xing et al. 2017) are able to perform some limited open-ended conversations. However, since they do not use explicit knowledge bases and do not perform inference, they often suffer from generic and dull responses (Li et al. 2017a; Xing et al. 2017).

A number of more recent works (Eric et al. 2017; Ghazvininejad et al. 2018; Le et al. 2016; Long et al. 2017; Madotto et al. 2018; Young et al. 2018; Zhou et al. 2018) have proposed to use knowledge bases (KBs) to help generate responses for knowledge-grounded

¹ <http://www.alicebot.org/>.

conversation. Many modern chatbots (e.g., conversational search and question-answering systems) also have an explicit KB storing real-world facts to support information-seeking conversations and help users with product recommendations. Here, a real-world fact stored in KBs is often represented as a triple: (s, r, t) , which says that the source entity s and target entity t are linked by the relation r . For example, $(Chicago, CityOf, USA)$ means that Chicago (source entity) is a city of (relation) USA (target entity). However, a major weakness of these chat systems is that they do not learn new knowledge during conversation, i.e., their knowledge is *fixed* beforehand and *does not expand or update during the conversation process*. This seriously limits the scope of their applications as it is almost impossible for the initial KBs to contain all possible knowledge that the user may ask, not to mention that new knowledge appears constantly. Even though some existing systems can use very large KBs, these KBs still miss a large number of facts (knowledge) as we know KBs are inherently incomplete (West et al. 2014). It is thus highly desirable for dialogue systems to acquire new knowledge directly from the user utterances or by explicitly asking users questions while in use.

Since there is little work in this emerging area, this chapter presents and discusses works that aim to build continual conversational knowledge learning systems for chatbots. In the following sections of this chapter, we first discuss various knowledge learning opportunities in conversation that can be leveraged to facilitate continual conversational knowledge learning (see Sect. 3.1). Section 3.2 focuses on learning knowledge by extracting facts from the dialogue context (conversation history). Section 3.3 focuses on learning lexical knowledge interactively in dialogue. Section 3.4 presents two methods for interactive factual knowledge learning and inference in the conversational question-answering setup. Section 3.5 presents a method for *passive knowledge learning*, which can help acquire and learn factual knowledge from external resources (like Web) and enrich the KB. Finally, we conclude with a chapter summary in Sect. 3.6.

3.1 Opportunities for Knowledge Learning in Dialogues

There are many opportunities to learn new knowledge during an actual conversation, after the chatbot is deployed. We characterize three broad approaches here:

1. **Extracting facts from user utterances.** Given the conversation history, the chatbot can extract facts from user utterances [see Fig. 3.1a] and store them in its back-end KB. For example, while conversing about movies, if the user says “*I watched Forest Gump yesterday. The movie was awesome. Liked Tom Hanks’ performance very much.*”, the chatbot can extract new facts like $(Forest\ Gump, isa, movie)$ and $(Tom\ Hanks, performed_in, Forest\ Gump)$ from the utterance(s). Later, the chatbot can leverage or ground these facts in future conversations while answering questions like “*Who acted in Forest Gump?*” or generating a response to user’s utterance “*I’m feeling bored. Can you recommend a*

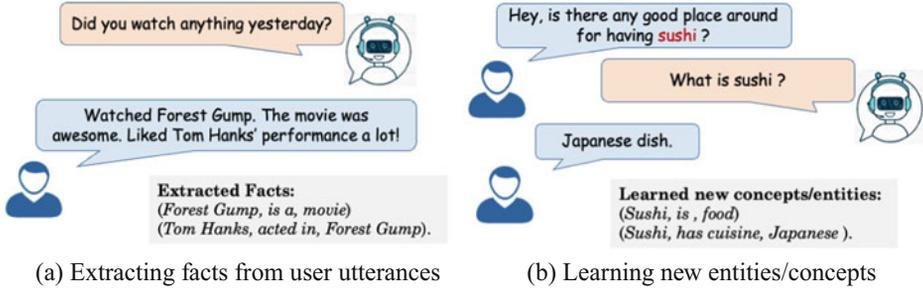


Fig. 3.1 Example dialogues showing various knowledge learning opportunities in dialogues

good movie?”. We will discuss more about this approach of conversational knowledge learning in Sect. 3.1.

2. **Ask questions to learn about unknown entities and concepts.** As unknown entities and concepts appear frequently in user utterances, the chatbot can ask clarification or information seeking questions to the user to acquire facts about new entities or concepts [see Fig. 3.1b]. For example, if the user says “*Is there any good place around for having sushi?*”, the chatbot can ask, “*Is sushi a food?*” or “*what is sushi?*”. On obtaining a response from the user, the chatbot can update its KB by incorporating the new concepts/entities (e.g. “sushi” in the said example) and associated facts about them. The process is formally known as *lexical knowledge acquisition* in dialogues and will be discussed more in Sect. 3.2.
3. **Ask and infer new facts.** When the chatbot cannot answer a user query, it can ask for some related supporting facts from the user through multi-turn conversation with the user and then, leverage the acquired facts and the ones already in KB to derive the answer of the query. We will discuss more about it in Sect. 3.3.

Note that the chatbot can not only ask *the current user* whom the chatbot is conversing with, but also *some other users* later because the current user does not know the answers of some questions. For example, a user asks the chatbot—“*Who is the president of the US ?*” today and say, the chatbot does not know the answer. Obviously, the chatbot should not ask the same question to the current user. But it can ask some other users later to obtain the knowledge so that it will be able to answer (in the future) when a later user asks the same question.

Besides the aforementioned approaches, the chatbot can also acquire knowledge by continually extracting facts from external resources like Web and populate its KB during conversation or off conversation. This is usually done in a *passive way* just like reading a book rather than asking others. We discuss such an approach in Sect. 3.6. Such pieces of learned real-world knowledge or facts can then be utilized automatically in future conversation sessions due to the continuous enrichment of its back-end KBs over time and thus, can directly

improve the performance of knowledge-grounded conversation modeling, conversational question answering, conversational search, interactive semantic parsing and many other conversational AI and IR applications.

3.2 Extracting Facts from Dialogue Context

The problem of extracting facts from dialogues can be viewed as an information extraction problem. Information Extraction (Banko et al. 2007; Fader et al. 2011; Hoffart et al. 2013; Mitchell et al. 2015; Wu and Weld 2010) is a widely studied topic in NLP that aims to extract facts as triples from sentences (or documents) and populate a KB with the extracted triples. In the conversation context, existing information extraction methods can be used by treating the conversation history as documents. However, there are also other associated challenges in dialogues such as co-reference resolution of entities and distinguishing facts from opinions, etc. that need to be solved along with the core information extraction task to achieve the goal.

Some approaches have been proposed to address the problem of knowledge extraction from conversation snippets and storing it into a KB for future use. We discuss two such approaches here.

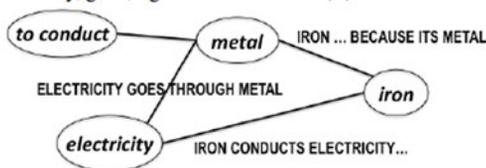
3.2.1 Learning Knowledge Graph from Dialogue

Constructing knowledge graph by extracting facts from the conversation context (or dialogue history) not only will be useful for knowledge-grounded conversation modeling, but also can help build external KBs for other downstream NLP tasks requiring real-world entities and commonsense facts. In this regard, Hixon et al. (2015) have studied the problem of how a question-answering (QA) system can learn domain knowledge from conversational dialogues. They have built a dialogue system called KNOWBOT that conducts dialogues about science questions, learns to relate concepts in each question to propositions in a fact corpus, stores new concepts and relations in a knowledge graph (KG), and uses the KG to derive the answers to the questions.

Figure 3.2 shows an example of a dialogue session and the corresponding knowledge graph construction in KNOWBOT. Here, the system presents its user with a question (S1 in Fig. 3.2) and prompts s/he to choose and explain his/her answer. In doing so, the system extracts relations [any semantic relationship between two concepts, such as *metal* and *iron* (U4 in Fig. 3.2)] that increase its confidence in the user's answer.

Fig. 3.2 An example dialogue session (top) and the corresponding knowledge graph (bottom) construction in KNOWBOT (Source Hixon et al. 2015)

S1: Here's the question: Which object is the best conductor of electricity? (A) a wax crayon (B) a plastic spoon (C) a rubber eraser (D) an iron nail. I'm not sure of the answer. What do you think the answer is?
U1: iron nail
S2: Why do you think the answer is (D) an iron nail?
U2: electricity goes through metal
S3: I know "metals are shiny and good conductors of electricity and heat," does that mean that the answer is (D) an iron nail?
U3: yeah
S4: Okay, great. What's the relationship between 'electricity' and 'iron', if any?
U4: iron conducts electricity because its metal
S5: Okay, great, I get it! The answer is (D) an iron nail



QA Task in KNOWBOT

The QA dataset for KNOWBOT consists of 107 science questions from the 4th grade New York Regents exam (Clark et al. 2014). Each question has four possible answers. Each of the four question-answer pairs is converted into a true/false question-answer statement using some pattern-based transformation rules. The fact corpus in KNOWBOT is built by collecting SCITEXT (Clark et al. 2014), which is a corpus of unlabeled true/false natural language sentences compiled from science textbooks, study guides and Wikipedia Science.

Given the dataset and fact corpus, each question-answer statement in the dataset is associated with a subset of true/false *support sentences* from SCITEXT, which is determined based on the degree of *positive word overlap* between the *question-answer pair* and the *support sentence*. For this, KNOWBOT extracts concepts from questions as well as SCITEXT sentences, and then use relations between the concepts to determine which question-answer statement is most highly aligned with a supporting SCITEXT sentence. The sentence alignment is computed as follows.

Let a *concept keyword* in a sentence or user utterance be any non-stopword of at least three characters. A *concept* is defined as a set of concept keywords with a common root, e.g. {*melts, melted, melting*}, {*heat, heats, heated*}, etc. The concepts appearing in a question-answer statement are referred to as *question concepts* and those appearing in a SCITEXT support sentence as *support concepts*. A *relation* is any pair of concepts that represents a semantic correspondence. For example, the words *because of* (relation) in the user utterance “*it's melting because of heat*” relates the concepts represented by *melt[ing]* and that by *heat*.

Then, the degree to which a *SCITEXT sentence* (say, S_j) supports a *question-answer pair* (say, Q_i) is measured by the sentence's alignment score $[\alpha(\cdot)]$ as follows:

$$\alpha(Q_i, S_j) = \frac{|R_{Q_i, S_j}|}{|C_{Q_i} \cup C_{S_j}|} \quad (3.1)$$

where, C_{Q_i} is the set of concepts in Q_i , C_{S_j} is the set of concepts in S_j and $|R_{Q_i, S_j}|$ is the number of relations between C_{Q_i} and C_{S_j} .

Knowledge Graph Construction in KNOWBOT

KNOWBOT grows a knowledge graph (KG) of commonsense semantic relations in an open conversational dialogue. Figure 3.3 shows the trace of the growth of a KG over a single dialogue. The KG is built in three levels: per utterance, referred to as *utterance-level knowledge graph* (uKG) [see Fig. 3.3a]; per dialogue, referred to as *dialogue-level knowledge graph* (dKG) [(Fig. 3.3b)] and globally over all dialogues, referred to as *global knowledge graph* (gKG). We discuss them below.

Utterance-level KGs. An *utterance-level knowledge graph* (uKG) is a fully connected graph whose nodes are all concepts in an utterance. KNOWBOT initially relates every pair of concepts in an utterance to build a uKG. This uKG is noisy because there may not be an intended relation between every pair of keywords in a user utterance. To remove noise, edges in uKG are aggressively pruned with two constraints: (i) *Alignment*, i.e., an edge can only relate a question concept to a support concept and (ii) *Adjacency*, i.e., edges cannot relate concepts whose keywords are adjacent in the utterance. After pruning, the remaining edges are added to the dialogue-level knowledge graph.

Dialogue-level KGs. Each dialogue focuses on a single question. KNOWBOT starts with an empty dialogue-level knowledge graph (dKG). After each user turn, edges from that turn's uKG are added to the dKG, and KNOWBOT re-scores each of the four answers using $\alpha(\cdot)$ where the set of relations R_{Q_i, S_j} is exactly the set of edges in the dKG. The dialogue

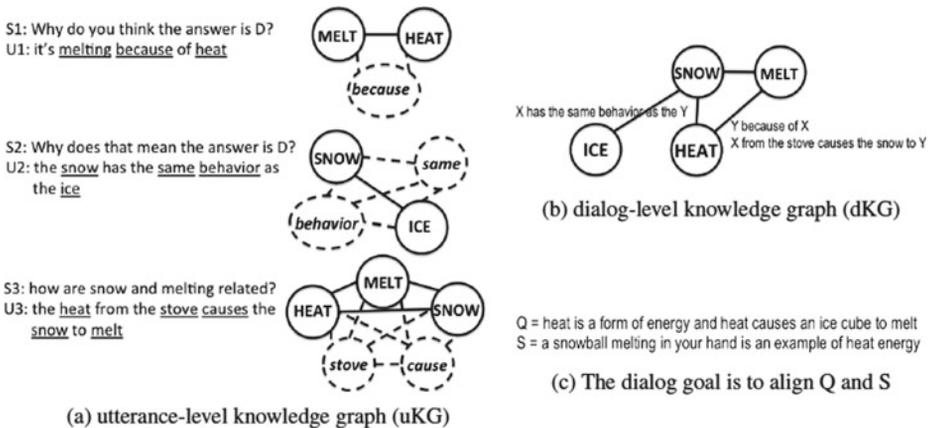


Fig. 3.3 Progressive knowledge graph construction in a given dialogue session by KNOWBOT (Source Hixon et al. 2015)

successfully terminates when the user’s answer has the highest $\alpha(\cdot)$, indicating the “missing knowledge” has been successfully provided by the user.

Global knowledge graph. The global knowledge graph (gKG) includes every relation learned from every KNOWBOT dialogue. Upon dialogue termination, the dKG is added to the gKG which stores relations acquired from all dialogues.

Interaction Strategies for Knowledge Acquisition

KNOWBOT always starts by asking users for their answers and if a new support sentence is found, it immediately presents the sentence to the user for confirmation. User provides explanation in response to a system prompt. Two types of dialogue strategies are used here: (1) a *user initiative strategy*, where the system asks open-ended questions to prompt the user for new explanations (e.g., line S2 in Fig. 3.2) and (2) a *mixed-initiative strategy*, where the system utilizes focused prompts (line S4 in Fig. 3.2) to acquire knowledge about potentially related concepts.

KNOWBOT chooses what pair of concepts to ask about based on how discriminative they are. The most discriminative concepts are the pair of question and support concepts that (1) does not already have an edge between them, (2) satisfies the alignment constraint for the user’s answer and (3) satisfies the alignment constraint for the fewest alternative answers.

3.2.2 Knowledge Acquisition in a Rule-Based Dialogue System

The work in Liu and Mei (2020) explored the aforementioned idea in rule-based dialogue systems. The authors proposed a system called *Knowledge Acquisition in Dialogues* (KAD) that learns by extracting information from user utterances and by asking the user. The acquired knowledge is expressed as triples, (e_1, r, e_2) , meaning that entities e_1 and e_2 have the relation r . For example, $(Miami, city-in, USA)$ means “*Miami is a city in the USA*”. KAD assumes that all relations are *known* and are in the knowledge base (KB). It thus learns only new instances of the relations, and new entities and their properties.

Learning Using Distillation Patterns

A rule-based dialogue system works by matching a user utterance u with a rule in the system and the system then generates a response for the user based on the rule. In the KAD system, a NER (named-entity recognition) system is first applied to identify the entities in u before performing rule matching. Further, the system also learns new knowledge from u in the background by instantiating knowledge distillation patterns (KDPs) attached to the rule with the matching results and then incorporates the knowledge into the KB. Each instantiated KDP is a piece of knowledge.

Some KDPs are attached to each rule to distill or extract knowledge from user utterances that match the rule. A rule with attached KDPs is specified as follows:

(p, F, B) , where F and/or B can be empty.

- p is a pattern to be matched with the user utterance.
- F is a set of KDPs representing fact triples $\{f_1, f_2, \dots, f_n\}$ that are true when instantiated with the matching results of the rule and the user utterance u , and can be added to the KB if they are not already there without confirming with the user. For example, we have the following enhanced rule (a rule with KDPs) for a hotel application:

p : [* stayed in X at Y]

F : (X, is-a, hotel), (X, has-address, Y)

If the user said “*I stayed in the Holiday Inn at 150 Pine Street last night.*” This utterance matches the pattern p of the above rule. Two pieces of knowledge (called *candidate facts*) from F should be true and added to the KB, which are

(*Holiday Inn, is-a, hotel*) and

(*Holiday Inn, has-address, 150 Pine Street*).

However, before adding, we need to make sure that they are correct via cross-verification.

Cross-verification: As a piece of knowledge or candidate fact f_i from the user may not be correct or trustworthy, we perform cross-verification by asking some other users of the system whether it is correct. However, in order to ask about f_i , we need to know what question to ask. The question is also attached to the relation *is-a* or *has-address* in the KB. For the two example facts above in F , the cross-verification questions are “*Is X a hotel ?*” and “*Is there a X hotel at Y ?*” respectively. If the cross-verification questions get affirmative answers from some other users, the two candidate fact triples can be added to the KB; otherwise the corresponding fact triple (knowledge) is discarded.

- B is a set of beliefs $\{b_1, b_2, \dots, b_m\}$ that may or may not be true. The beliefs are used because in some cases, we are not sure whether a triple can be implied by the utterance matching p . In this case, for each b_i , the system needs to ask the user to confirm its truth value. Each b_i consists of two specifications (m_i, A_i) , where m_i is the main belief and A_i is a set of auxiliary facts $\{a_1, a_2, \dots, a_k\}$ that are true and should be added to the KB if m_i is true and is added to the KB. They represent the *properties* and values of the entities involved in the main belief. m_i is also implicitly associated with two questions: a confirmation question to ask the current user and a cross-verification question to ask other users to ensure correctness. Again, the questions are attached to the relations in the KB. More details can be found in the original paper.

Note that when we say a triple is added to the KB, we mean that we only add the missing information. For example, if the entities are already in the KB, we only add the relation. In order to know whether an entity is already in the system, we need to deal

with people using different names to refer to the same entity. An existing entity linking system can be used to resolve the confusion.

Knowledge Base (KB)

KB is a knowledge graph consisting of a set of triples $\{(e_1, r, e_2) \mid e_1, e_2 \in E, r \in R\}$, involving entities and their relations. E is the entity set and R is the relation set of the KB. Each entity $e_i \in E$ has an *is-a* type hierarchy and each *type* (e.g., hotel) has a set of properties (e.g., address, parking, etc.) that will be used by each instantiated instance of the type.

Each relation is attached with a factual question q^f that the system asks the current user about a particular relation instance and a cross-verification question q^v that the system asks some other users whether the relation instance is true.

A special treatment is needed for property relations because we want to ask for property values of a known entity. Needless to say that each entity instance of a type inherits the properties of the type. For example, for the property relation *has-address* for the *hotel* type, we can have

q^f : What is the address of X ?

q^v : Is there a X hotel at this address, Y ?

where X is an entity instance, e.g., Holiday Inn. With the attached factual question, the system can ask for the property value of a particular instance of an entity type, e.g., “*what is the address of the Holiday Inn ?*” The system can learn from the answers of the user. Likewise, cross-verification questions can be asked by using q^v , e.g., “*Is there a Holiday Inn at this address, 150 Pine Street ?*”

For non-property relations, q^f is a confirmation question. For example, for the *is-a* type relation, we may have the following questions (q^f and q^v happen to be the same in this case):

q^f : Is X a hotel?

q^v : Is X a hotel?

To make this technique practically useful, a few issues have to be resolved: (1) not asking too many questions to the same user; (2) ordering the questions that need to be asked. (3) asking a user information that s/he is likely to know or identifying the right user(s) to ask each question (e.g., asking doctors about medicine); (4) identifying which questions are factual and which ones are subjective (e.g., whether a movie is good); (5) deciding the number of users that should be asked about a question for the relation/fact to be deemed reliable; and (6) prioritizing the concepts to learn more about (e.g., many other users are likely to ask questions about a new celebrity that one user mentions, than one user’s imaginary friend).

Finally, it is also important to know that some questions may not be reasonable to ask a user, e.g., “*Is there a Holiday Inn at this address, 150 Pine Street?*” as few people would know unless the person lives nearby or has just stayed in the hotel. For such types of questions, some external knowledge base or directory may be used to confirm the knowledge.

3.3 Lexical Knowledge Acquisition in Dialogues

Most of the dialogue systems are deployed in practice with pre-compiled KBs (having *limited* vocabulary) that are not updated during the conversation. Thus, these dialogue systems become clueless on encountering *unseen* concepts or terms from end users and perform poorly. Several researchers have attempted to solve this problem. They proposed to enable dialogue systems to interactively learn about unknown concepts or terms (when encountered) from users to enrich their back-end KBs.

Otsuka et al. (2013) and Ono et al. (2017, 2016) have studied the problem of *lexical knowledge acquisition* in dialogues, where the dialogue system aims to acquire the *attributes* or *ontological categories* of unknown terms from users through multi-turn dialogue. Figure 3.4 shows an example of lexical knowledge acquisition, where the system asks user questions about unknown terms (e.g. “*Toyo*” and “*Osteria Liu*” in the figure) to acquire knowledge. Kollar et al. (2013), Padmakumar et al. (2018) and Thomason et al. (2017) have studied the problem of grounding object or location descriptions through interactive dialogue with users. The dialogue system in Kollar et al. (2013) is relatively simple. For example, it directly asks the room number of a person’s office. The systems in Padmakumar et al. (2018) and Thomason et al. (2017) are more sophisticated, which use reinforcement learning to learn a query policy.

In the following section, we mainly focus on discussing some representative works in the topic. First, we discuss how the dialogue system can generate questions for acquiring attributes of an unknown concept appearing in a dialogue (Otsuka et al. 2013) (Sect. 3.3.1). Next, we focus on how lexical knowledge can be acquired through implicit confirmations over dialogue (Ono et al. 2016, 2017) rather than asking explicit verification questions (Sect. 3.3.2).

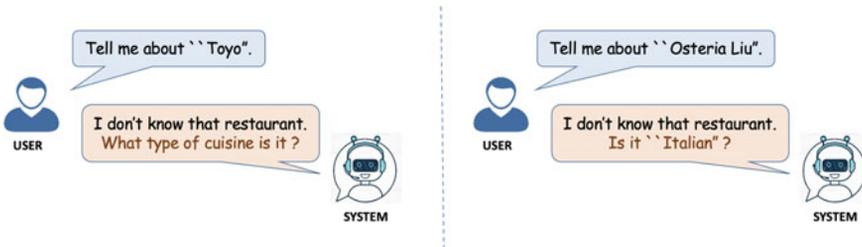


Fig. 3.4 Example showing lexical knowledge acquisition, where the system asks user questions to acquire knowledge about unknown terms (Source adapted from Otsuka et al. 2013)

3.3.1 Generating Questions for Acquiring Attributes of Unknown Concepts

To acquire the attribute of an unknown concept or term, the dialogue system can either ask a Wh-question (e.g., “*what type of cuisine is it?*”) or a specific question (e.g., “*Is it Italian?*”) as shown in Fig. 3.4. However, asking specific questions are more preferable than Wh-questions in the dialogue setting, as they can narrow down variations of the following user responses. Also, they lead to better Automatic Speech Recognition (ASR) performance of the response and reduce the risk of encountering further unknown terms in the user’s response.

In this regard, Otsuka et al. (2013) have proposed a method for generating more specific questions to acquire attributes of unknown terms based on an appropriately distributed *confidence measure* (CM). The overall idea is as follows. The system first predicts the attribute of an unknown concept, and then uses the CM of the prediction to decide whether to ask a simple Wh-question or a more specific question. To study the problem, they formulate a restaurant database retrieval task, where they consider the *restaurant names* to be the *unknown* terms and the *cuisine types* as their *attributes*. They assume that the system already knows that the unknown terms in user’s utterance are restaurant names, but is unaware of what kind of restaurant it is (i.e., their cuisine type).

For example, in Fig. 3.4, the system knows “*Toyo*” and “*Osteria Liu*” are restaurants and attempts to formulate questions to acquire their cuisine types (as attributes) from the user. In doing so, the system first predicts their cuisine types and then, formulates a more specific question (e.g., “*Is it Italian?*” in case of “*Osteria Liu*”) if the prediction is believed to be a highly confident one. Otherwise (when the system has no clue), it asks a Wh-question about the restaurant’s cuisine type (e.g., “*what type of cuisine is it?*” in case of “*Toyo*”).

Question Generation based on CM. The type of questions is determined based on a well-defined CM. The CM is estimated for each cuisine type $c_i \in C$ in the target restaurant database, where $|C|$ is the total number of possible cuisine types in the database. Figure 3.5a

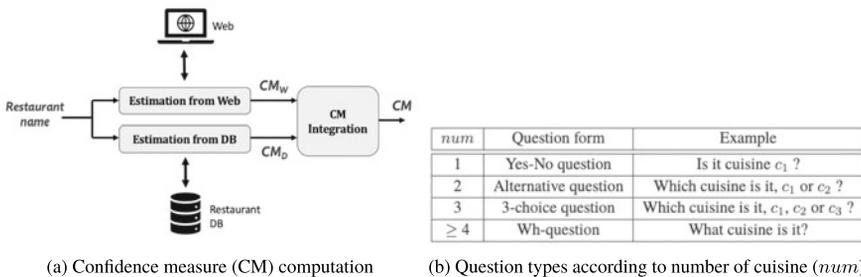


Fig. 3.5 Question generation based on the confidence measure (Source adapted from Otsuka et al. 2013)

shows the confidence measure computation process and Fig. 3.5b shows various question types to be generated based on a parameter— num computed using CM. Here, num is the *number of cuisine types* that should be included in the question.

If the system obtains one cuisine type that it is very confident about (has a high CM), it generates the most specific question (i.e., a Yes-No question). See the case where $num = 1$ in Fig. 3.5b. If CM is low (i.e. unreliable cuisine types are obtained), the system generates questions including several cuisine types (see the questions for $1 < num < 4$ in Fig. 3.5b). If $num \geq 4$, asking a verification question involving a large number cuisine types is not user friendly and thus, the system asks a Wh-question, considering it is clueless in the estimation.

CM is computed utilizing information from both the Web and the restaurant database included with the system. Given a restaurant name (unknown term here), two types of CMs are calculated based on (1) character and word distributions of the *restaurant name* in the target database (CM_D) and (2) frequency of occurrence of the *restaurant attributes* on Web pages (CM_W). Then, these two CMs are combined to compute the final CM score (see Fig. 3.5a).

CM_D is calculated using a supervised learning model trained with a list of (restaurant name, cuisine) pairs in the restaurant database. The intuition here is that some cuisine types can be estimated from restaurant names on the basis of their character types or typical character sequences they contain. Otsuka et al. (2013) trains a maximum entropy (ME) model as a classifier for CM_D , where the feature vectors obtained from each restaurant name are computed based on *Character n-grams* ($n = 1, 2, 3$), *Words* and *Character types* and mutual information are used to perform feature selection.

A restaurant’s name often appears with its cuisine type on Web pages. Following this intuition, CM_W is calculated based on the frequency of occurrence of restaurant attributes on the Web. First, given the restaurant name (as a search query), the system retrieves 20 pages (set as limit) per query using the Yahoo! search API. Next, the system computes the frequency of each cuisine type c_j in the i^{th} Web page (ranked by the Web search API) and sums up the frequency through all the obtained pages to calculate the posterior probability $P_{freq}(c_j)$ of c_j . Finally, $P_{freq}(c_j)$ is scaled over all cuisine types to obtain the $CM_W(c_j)$ score.

The computed CM_D and CM_W are then integrated using logistic regression to obtain the final CM score, which is then used to compute the number of cuisine types (num) as:

$$num = \min(n) \text{ s.t. } \sum_{j=1}^n CM(c_j) > \theta \quad (3.2)$$

where, $CM(c_j)$ is the confidence measure for cuisine type c_j in *its descending order* and θ is a threshold determined based on the distribution of $CM(c_j)$. Here, if only the $CM(c_1) > \theta$ (i.e., $n = 1$), the system generates a specific question involving only c_1 , whereas if $\sum_{j=1}^4 CM(c_j) < \theta$ (i.e., $n = 4$), the system considers itself to be less confident and generates a Wh-question.

3.3.2 Knowledge Acquisition via Implicit Confirmations over Dialogues

As discussed in Sect. 3.3.1, the dialogue system can ask specific questions to the user to acquire knowledge about unknown terms during conversation. However, asking for such explicit confirmation requests repetitively to acquire lexical knowledge can degrade user's experience. In contrast, lexical knowledge acquisition through implicit confirmation requests (Ono et al. 2016) can be a more desirable approach as it maintains the natural flow of the conversation.

Ono et al. (2017) have studied the problem of acquiring lexical knowledge through implicit confirmation in dialogues with multiple different users in a closed-domain Japanese language chatbot targeting the food and restaurant domain. The goal here is to acquire the categories of foods that the system does not know. They assume that the system can identify a food name in the user's input (even if it is not in the system's vocabulary) by using methods such as named entity recognition and the category of an unknown term predicted with an existing methods (Ono et al. 2016; Otsuka et al. 2013). Based on this setting, Ono et al. (2017) have developed an approach that makes implicit confirmation requests with an unknown term's predicted category and uses the user's response to decide if the category is correct or not.

Figure 3.6a and b show examples of various explicit and implicit confirmation requests in dialogues respectively. For example, in Fig. 3.6a, the system estimates the category of restaurant “*nasi goreng*” as *Indonesian* using the approach in Otsuka et al. (2013) and asks an explicit question or confirmation request. This can degrade user's experience when the predicted category is incorrect or obvious to the user. On the other hand, the implicit confirmation requests made in Fig. 3.6b maintains a natural flow.

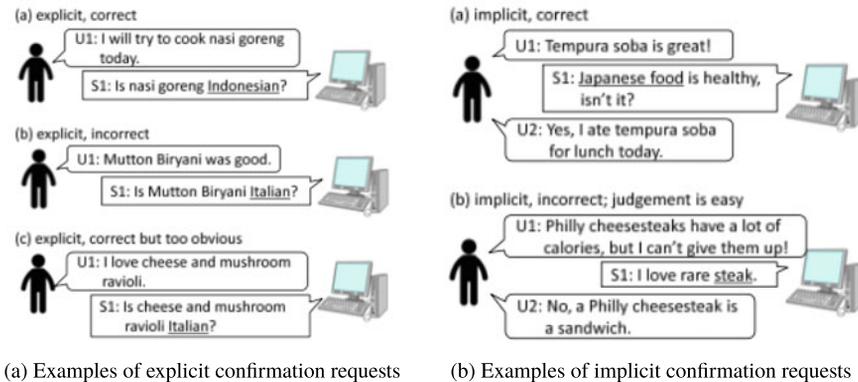


Fig. 3.6 Examples showing explicit and implicit confirmation requests in dialogues (Source Ono et al. 2017)

However, determining the correctness of a predicted category in implicit confirmation requests can be challenging. For example, considering the dialogue below, the user’s response can be misleading to the system (as pasta is not a Mexican food, rather it’s an Italian food).

User : I ate pasta yesterday.
 System : Sometimes I like to eat Mexican food.
 User : Me too.

Thus, accurately determining the correctness of the predicted categories becomes essential to maintain the veracity of the system’s KB. Ono et al. (2017) computed a confidence score of the predicted category being correct from the responses of multiple users to the same implicit confirmation requests and leverage it for solving the problem, as described below.

Calculating confidence score using multiple confirmation requests. Let $p_i(w, c)$ denotes the probability that a predicted category c of an unknown term w is correct after a single implicit confirmation request is made. Here, i denotes the i^{th} response to implicit confirmation requests. The goal is to obtain a confidence score $Conf(w, c)$ denoting how likely category c of the unknown term w is correct based on the replies to the requests from n different users. Figure 3.7 shows an overview of the confidence score computation process. The process starts with $i = 1$ and proceeds as follows:

1. Generate an implicit confirmation request containing a predicted category c for user i after an unknown term w appears.
2. Obtain the probability $p_i(w, c)$ from the implicit confirmation sub-dialogue with user i . $p_i(w, c)$ is computed using a machine learning model that employs various linguistic features (e.g., whether the response includes an expression affirmative (or negative) to

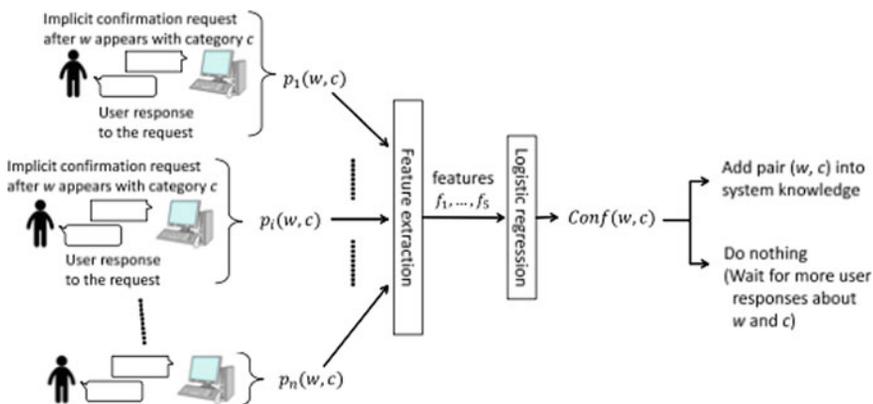


Fig. 3.7 Calculating confidence score using multiple confirmation requests (Source Ono et al. 2017)

- the context, whether the response uses the category name appeared in the context, etc.) involving expressions appearing in the user response and its context.
3. Extract features from $\{p_1(w, c), \dots, p_i(w, c)\}$ and calculate the confidence score $Conf(w, c)$. This is formulated as a regression task, where the probabilities are used as input and $Conf(w, c)$ is computed using various statistical features (like *Average*, *Max*, *Min*, etc.) involving these probabilities.
 4. If $Conf(w, c)$ exceeds a pre-determined threshold, c is regarded as correct and is acquired as knowledge. Otherwise, increment i and go to Step 1, and generate one more implicit confirmation with c to another user after the unknown term w appears.

3.4 Interactive Factual Knowledge Learning and Inference

Interactive factual knowledge learning and inference focuses on learning and inferring new facts from users interactively in multi-turn dialogues. Mazumder et al. (2019, 2020) have explored this task in the conversational Question-Answering (QA) setting. The idea of interactive factual knowledge learning can be explained as follows. In a given dialogue session, when the user asks the chatbot either a WH-question (e.g. “*In what country is Boston located?*”) or a verification question (e.g. “*Is Obama a citizen of USA?*”), the chatbot attempts to answer the question. However, considering the chatbot has to operate in the open-world setting (in practice), it’s quite natural that the query entity (say, “*Boston*” in the WH-question example) and/or the query relation (say, “*located in*” in the WH-question example) may be unknown (i.e., the entity and/or relation do not exist in the chatbot’s KB) making the system unable to answer the questions. Moreover, even if they are known, the existing facts in the KB may not be sufficient to derive the answer (West et al. 2014). Thus, the chatbot needs to acquire additional knowledge from the user before it can reason over its KB and infer the answer.

Mazumder et al. (2019, 2020) dealt with the above situation by enabling the chatbot to interact with the user to acquire additional knowledge (which is called **supporting facts**) and use it along with the existing knowledge in the KB for inference. We discuss two such systems, namely *Continuous and Interactive Learning of Knowledge* (CILK) and *Interactive Knowledge Acquisition and Inference* (IKAI) in the following subsections. CILK leverages the situation when the system is unable to answer a user’s WH-question and IKAI enables interactive factual knowledge learning in the verification QA setting. Both systems focus only on developing the core knowledge learning method. They do not study other dialogue functions like response generation, semantic parsing, fact extraction, entity linking, etc. For example, CILK works only with structured queries $(h, r, ?)$, e.g., $(Boston, LocatedInCountry, ?)$ meaning “*In what Country is Boston located?*.” It assumes that a semantic parser is available that can convert natural language queries from users (issued in conversational QA setting) into query triples and an information extraction tool like OpenIE (Angeli et al. 2015) is employed to extract facts as triples (h, r, t) from user’s utterances.

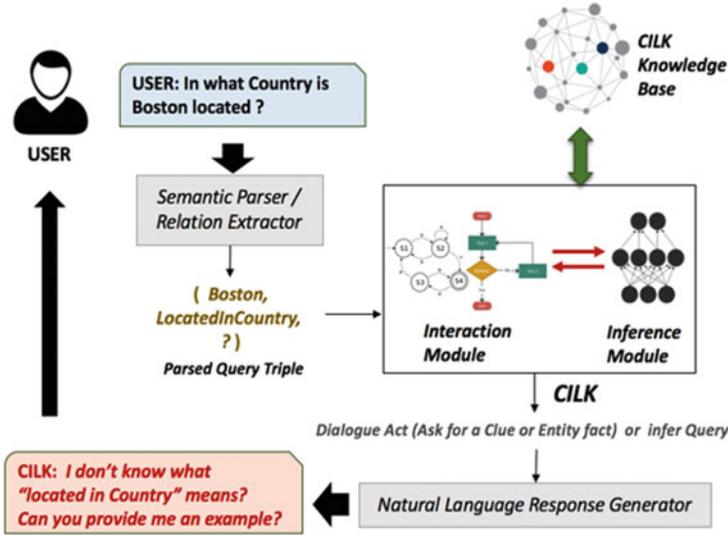


Fig. 3.8 Integration of CILK knowledge Learning framework with its peripheral components

Figure 3.8 shows how CILK fits in a dialogue system and operates in a given dialogue session with the help of its additional peripheral components.

3.4.1 The CILK Framework

Given a user WH-query $(h, r, ?)$ [or $(?, r, t)$], where r and h (or t) may not be in the KB (i.e., *unknown*), the goal of CILK (Mazumder et al. 2019) is two-fold: (i) *answering the user query* or *rejecting the query to remain unanswered* in the case when the correct answer is believed to not exist in the KB and (ii) *learning/acquiring some knowledge (facts) from the user to help the answering task*. The focus is on the setting where the query cannot be answered *directly* with the current KB and needs inference over existing facts. The authors further distinguish two types of queries: (1) *closed-world queries*, where h , t and r are known to the KB, and (2) *open-world queries*, where either one or both h (or t) and r are unknown to the KB.

It is easy to see that the problem is essentially a *lifelong learning* problem (Chen and Liu 2018) (discussed in Chap. 2), where each query to be processed is a task and the knowledge gained is retained in the KB. To process a new query/task, the knowledge learned and accumulated from the past queries can be leveraged. For each new *open-world* query, CILK works in two steps:

Step 1—Interact with the user: It converts an open-world query (2) to a closed-world query (1) by asking the user questions related to h (or t) and r to make them known to the KB (added to the KB). The reason for the conversion will be clear below. The user answers, called *supporting facts* (SFs), are the new knowledge to be added to the KB. This step is also called *interactive knowledge learning*. Note, closed-world queries (1) do not need this step.

Step 2—Infer the query answer: It solves the closed-world query (1) by inferring the query answer. The main idea is to use each entity e in the KB to form a candidate triple (h, r, e) [or (e, r, t)] to be scored. The entity e with the highest score is predicted as the answer of the query. Scoring each candidate is modeled as a *knowledge base completion* (KBC) problem (Bordes et al. 2011; Lao and Cohen 2010). KBC aims to infer new facts (knowledge) from existing facts in a KB and is defined as a *link prediction* problem: Given a query triple, $(e, r, ?)$ [or $(?, r, e)$], it predicts a tail entity t_{true} [or head entity h_{true}] that makes the query triple true.

Figure 3.9 shows an example. CILK acquires supporting facts SF1 and SF2 to accomplish the goal of *knowledge learning* and utilizes these pieces of knowledge along with existing KB facts to answer the user query (i.e., to infer over the query relation “*LocatedInCountry*”). CILK aims to achieve these two sub-goals. The new knowledge (SFs) is added to the KB for future use.² In Fig. 3.9, $(London, LocatedInCountry, UK)$ is a clue of query relation “*LocatedInCountry*” and $(Harvard\ University, UniversityLocatedIn, Boston)$ is an entity fact involving query entity “*Boston*”. In this interaction process, CILK decides and asks questions to the user for knowledge acquisition in multiple dialogue turns (see Fig. 3.9).

USER: $(Boston, LocatedInCountry, ?)$ “In what Country is Boston located?” [Query]
CILK: I do not know what “**located in Country**” means? Can you provide me an example? [Ask for Clue]
USER: $(London, LocatedInCountry, UK)$. “London is located in UK.” [SF1]
CILK: Got it. Can you tell me a fact about “**Boston**”? [Ask for Entity Fact]
USER: $(Harvard\ University, UniversityLocatedIn, Boston)$. “Harvard university is located in Boston.” [SF2]
CILK: $(Boston, LocatedInCountry, USA)$ “Boston is located in USA.” [Answer]

Fig. 3.9 An example of interactive learning and inference. Note that CILK only works with triples. Each triple above is assumed to be extracted from the sentence after it. *Ask for Clue* and *Ask for Entity Fact* are interaction query types. SF denotes supporting fact

² The inferred query answer is not added to the KB as it may be incorrect. But it can be added if verified in a multi-user environment through cross-verification.

The CILK Knowledge Learning Components

As discussed above, given a query $(e, r, ?)$ [or $(?, r, e)$]³ from the user, CILK interacts with the user to acquire supporting facts to answer the query. Such an interactive knowledge learning and inference task is realized by the cooperation of three primary components of CILK: **Knowledge base** (KB) \mathcal{K} , **Interaction Module** \mathcal{I} and **Inference Model** \mathcal{M} . The interaction module \mathcal{I} decides whether to ask or not and formulates questions to ask the user for supporting facts. The acquired supporting facts are added to the KB \mathcal{K} and used in training the Inference Model \mathcal{M} which then performs inference over the query (i.e., answers the user query).

Inference Model (\mathcal{M}) of CILK. Given a query q , the Inference Model \mathcal{M} attempts to infer q by predicting the answer entity from \mathcal{E} in the KB. In particular, it selects each entity $e_i \in \mathcal{E}$ and forms $|\mathcal{E}|$ number of candidate triples $\{d_1, \dots, d_{|\mathcal{E}|}\}$, where d_i is of the form (e, r, e_i) for a tail query [or (e_i, r, e) for a head query] and then score each d_i to quantify the relevancy of e_i of being an answer to q . The top ranked entity e_i is returned as the predicted answer of q . We deal with the case of query rejection by \mathcal{M} later. Mazumder et al. (2019) have used the neural knowledge base embedding (KBE) approach (Bordes et al. 2013, 2011; Yang et al. 2015) to design \mathcal{M} .

Rejection in KB Inference. For a query with no answer entity existing in \mathcal{K} , CILK attempts to reject the query from being answered. To decide whether to reject the query or not, CILK maintains a **threshold buffer** \mathcal{T} that stores entity and relation specific prediction thresholds and updates it continuously over time.

To minimize the rate of user interaction and justify the knowledge acquisition process, CILK uses a **performance buffer** \mathcal{P} to store the performance statistics of CILK in past dialogue sessions. It uses Mean Reciprocal Rank (MRR) to measure the performance of \mathcal{M} . In particular, $\mathcal{P}[e]$ and $\mathcal{P}[r]$ denote the avg. MRR achieved by \mathcal{M} while answering queries involving e and r respectively, evaluated on the validation dataset D_{vd} . At the end of each dialogue session, CILK detects the set of bottom $\rho\%$ query relations and entities in \mathcal{P} based on the MRR scores evaluated on the validation dataset. These sets are called the **diffident** relation and entity sets respectively *for the next dialogue session*. If the query relation and/or entity issued in the next session belongs to the *diffident* relation or entity set, CILK asks the user for supporting facts.⁴ Otherwise, it proceeds with inference, answering or rejecting the query.

³ Either e or r or both may not exist in the KB.

⁴ Note, if (unknown) e or r appears the first time in a user query, then it cannot be in the diffident set. But the system has to ask the user a question by default.

3.4.2 The IKAI Framework

Different from CILK, IKAI (Mazumder et al. 2020) focuses on interactive factual knowledge learning and inference in verification dialogues, i.e. when the user asks a factual verification question (e.g. “*Is Obama a citizen of USA?*”) to the chatbot and chatbot is unable to validate the fact in the question. Like CILK, IKAI also interacts with the user to acquire some supporting facts and use them to answer the verification question. For example, when the user asks “*Is Obama a citizen of USA?*” [see Fig. 3.10], the chatbot first looks-up in its KB. If the fact is not in the KB, it can try to infer the answer from the existing KB facts. However, this becomes infeasible if any of the entities (“*Obama*” and “*USA*”) or relation (“*CitizenOf*”) is *unknown*. The chatbot thus asks the user to provide some clues or related knowledge to help it infer the answer, similar to CILK.

The knowledge inference task is formulated as the following *Knowledge Base Completion* (KBC) problem: Given a verification (query) triple $(s, r?, t)$, it predicts whether the entities s and t can be linked by relation r . Here we use $?$ to indicate a query triple. Existing KBC methods (Bordes et al. 2011, 2013; Lao et al. 2011; Nickel et al. 2015) typically makes the *closed-world assumption* that s , r and t are all *known* to exist in the KB. This assumption is a serious limitation because those facts involving entities and/or relations not already in the KB cannot be used in reasoning by existing KBC methods. Like CILK, IKAI also removes this assumption and allows s , r and/or t not to exist in the KB and formulates the inference problem as *open-world knowledge base completion* (OKBC) problem. This involves following two subtasks:

Task 1: Interactive acquisition of supporting facts, i.e., formulating an inference strategy to ask the user suitable questions to convert an OKBC query (in which any of s , r and t does not exist in the KB) to a KBC query (where all of s , r and t exist in the KB). Like CILK, the acquired user answers (which are *supporting facts*) are added to the KB, making the OKBC query reduced to a KBC one.

USER:	<i>(Obama, CitizenOf?, USA)</i> “Is Obama a citizen of USA?”	[Query]
IKAI:	I do not know what “ CitizenOf ” means? Can you provide me an example?	
		[Ask for Clue]
USER:	<i>(David Cameron, CitizenOf, UK)</i> . “David Cameron is a citizen of UK.”	[SF1]
IKAI:	Got it. Can you tell me how “ Obama ” and “ Honolulu ” are related?	[CLQ]
USER:	<i>(Obama, BornIn, Honolulu)</i> . “Obama was born in Honolulu.”	[SF2]
IKAI:	Got it. Can you tell also me how “ Honolulu ” and “ Hawaii ” are related?	
		[MLQ]
USER:	<i>(Honolulu, CapitalOfState, Hawaii)</i> . “Honolulu is the state capital of Hawaii.”	
		[SF3]
IKAI:	<i>(Obama, CitizenOf?, USA)</i> holds. “Yes, Obama is a US citizen.”	[Answer]

Fig. 3.10 An example of interactive learning and inference. Like CILK, IKAI only works with triples. Each triple above is assumed to be extracted from the sentence after it. *Ask for Clue*, *CLQ* (Connecting Link Query) and *MLQ* (Missing Link Query) are interaction query types

Task 2: Knowledge inference, i.e., building a predictive model based on the supporting facts and the knowledge already in the KB to infer whether the converted KBC query is true or not, which answers the original user query.

Figure 3.10 shows an example. IKAI acquires the *supporting facts* SF1, SF2, SF3 to accomplish the *interactive knowledge acquisition* (Task 1) and then utilizes these pieces of knowledge along with existing KB facts to *infer* over the query relation “*CitizenOf*” to answer the user query (Task 2). The new knowledge (which are supporting facts) is also added to the KB for future use.

Knowledge Learning in IKAI. IKAI is implemented with a *Finite State Machine* (FSM) and a *Predictor*. The FSM dynamically formulates an inference strategy for a given OKBC query. While executing the strategy, IKAI processes the query and acquires the supporting facts by asking the user. The supporting facts being added to the KB transforms an OKBC problem to a KBC one (i.e., all s , r and t become *known* to the KB). The *Predictor* is then invoked for knowledge inference to solve the *transformed* KBC problem.

Different from CILK, IKAI uses a *path-ranking* (PR) approach (Gardner et al. 2014; Lao et al. 2011, 2015, Neelakantan et al. 2015) to build the Predictor. PR-based methods enumerate relation paths (sequence of relations) between two entities in a KB (encoded as a multi-relation graph) and use those relation paths as features to train a binary classifier for predicting missing relations between them. As several paths can link two entities in a KB and the KB becomes more dense with addition of new knowledge, existing PR based methods suffer from scalability (high RAM consumption) and feature explosion (trains on an exponentially large number of features) problems. As a result, they are not directly adoptable for *online real-time* knowledge learning and inference. Thus, IKAI adopts a variant of PR-based method—*Context-aware Path Ranking* (C-PR) (Mazumder and Liu 2017) to build the *Predictor* that learns from fewer and more discriminative features meeting the real-time requirements while showing improved inference performance.

The FSM in IKAI works as follows. At each state S_i , FSM chooses an action to execute. An action can be a *processing action* or an *interaction action*. A processing action advances the reasoning (like searching query entities and relation in the KB, extracting path features). An interaction action [e.g., Asking for a Clue, a Connecting Link Query (CLQ) and a Missing Link Query (MLQ)] asks and acquires knowledge from the user. Here, IKAI uses CLQ to ask user a question for connecting a new entity with an existing entity in the KB and uses MLQ for filling up missing relations between two entities (not having any direct relation between them) in the KB.

After an action is executed, FSM goes to a new state S_{i+1} . Through this process, an OKBC query gets gradually transformed into a KBC one. Additionally, the process also performs path feature extraction (processing actions) for a clue triple or a query triple for training the Predictor (see Fig. 3.10). When FSM transits into its final state, the *Predictor* is invoked to train and/or infer the reduced KBC query to answer the original user question.

3.5 Learning New Knowledge from External Sources

A dialogue system may also learn knowledge through other means and from other sources rather than only from the person that it converses with during a dialogue. Like what we humans do, one important way of learning is through reading text documents, such as books, news articles and the Web. A lifelong learning system should also do the same and extract knowledge from them. This area of research is called *information extraction* that aims to extract structured knowledge from unstructured text. An extensive research has been done on information extraction from the Web (Chang et al. 2006; Mirza et al. 2018; Nayak et al. 2021; Niklaus et al. 2018; West et al. 2014; Wise et al. 2020; Zhang and Balog 2020). However, few systems can do this continuously. Here we discuss one work called NELL (*Never-Ending Language Learner*) (Carlson et al. 2010; Mitchell et al. 2015) that reads the Web and extracts two types of factual knowledge. It is perhaps the only live continuous learning system that extracts information (or knowledge) from the Web.

NELL is a lifelong semi-supervised information extraction system. It uses a small number of labeled training examples for each of its learning tasks. The *input* to NELL consists of the following:

1. An ontology defining a set of target categories and relations to be learned (in the form of a collection of *predicates*), a handful of seed training examples for each, and a set of constraints that couple various categories and relations (e.g., *Person* and *Sport* are mutually exclusive),
2. Webpages crawled from the Web, which NELL uses to extract information, and
3. Occasional interactions with human trainers to correct some of the mistakes made by NELL.

With this input, the *goal* of NELL is two-fold.

1. Extract facts from the webpages to populate the initial ontology. Specifically, NELL continuously extracts the following two types of information or knowledge:
 - (a) *category* of a noun or noun phrase, e.g., Los Angeles is a *city*, Canada is a *country*, and New York Yankees is a *baseball team*.
 - (b) *relations* of a pair of noun phrases. For example, given the name of a university (say *Stanford*) and a major (say *Computer Science*), check whether the relation *hasMajor*(*Stanford*, *Computer Science*) is true.
2. Learn to perform the above extraction tasks, also called the *reading tasks*, better than yesterday.

To achieve these objectives, NELL works iteratively in an infinite loop, hence *never-ending* or *lifelong*. Each iteration performs two main tasks corresponding to the two objectives:

1. **Reading task:** The task is about reading and extracting the two types of information or knowledge from the Web to grow the KB of structured facts (or beliefs). Because of a huge number of possible candidate facts, NELL considers only those with the highest confidences as beliefs. This enables NELL to operate tractably and also to be able to add millions of new beliefs over many iterations.
2. **Learning task:** This task aims to learn to read better with the help of the accumulated knowledge in the updated KB and some coupling constraints. The training examples consist of a combination of human-labeled instances (a dozen or so labeled seed examples provided for each category and relation in NELL's ontology), labeled examples contributed over time through NELL's crowdsourcing website, a set of NELL self-labeled training examples corresponding to NELL's current/updated knowledge base, and a large amount of unlabeled Web text. The last two sets of training examples propel NELL's lifelong learning and self-improvement process.

3.5.1 NELL Architecture

NELL's architecture is shown in Fig. 3.11. There are four main components in NELL: data resources, knowledge base, subsystem components and knowledge integrator.

Data Resources. Since the goal of NELL is to continuously read webpages crawled from the Web to extract knowledge, webpages are thus the data resources.

Knowledge Base. Knowledge Base (KB) stores all the extracted knowledge that is expressed as beliefs. As mentioned above, two types of knowledge are stored in the knowledge base: instances of various categories and relations. A piece of knowledge can be a candidate fact or a belief. A candidate fact is extracted and proposed by the subsystem components, and may be promoted to a belief, which is decided by Knowledge Integrator.

Subsystem Components. NELL contains several subsystems (i.e., CPL, CSEAL, CMC, and RL), which are the extractors and learning components. In the reading phase, these subsystems perform extraction and propose candidate facts to be included in the knowledge base. In the learning phase, they learn based on their individual learning methods.

Knowledge Integrator. Knowledge Integrator (KI) controls the condition of promoting candidate facts into beliefs. It uses a set of hand-coded rules.

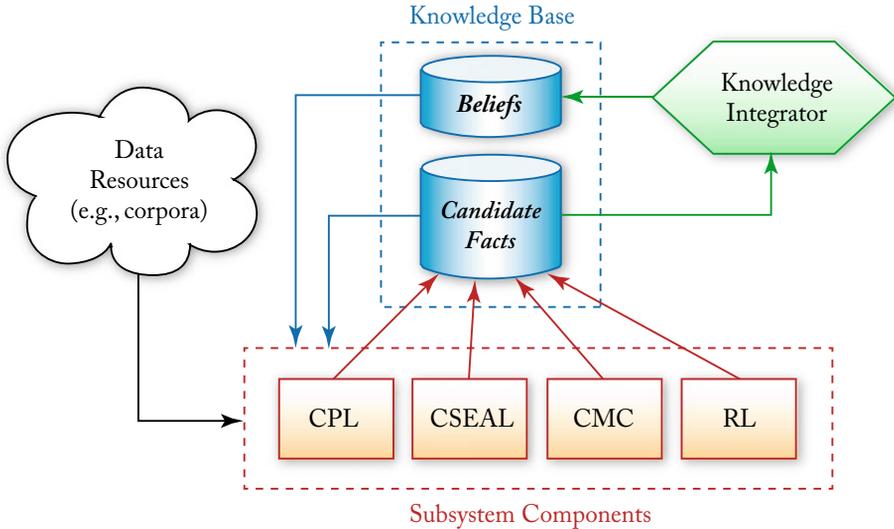


Fig. 3.11 NELL system architecture (Carlson et al. 2010)

3.5.2 Extractors and Learning in NELL

As we can see in Fig. 3.11, there are four major subsystem components that perform extraction and learning (Carlson et al. 2010). We discuss them below.

- **Coupled Pattern Learner (CPL):** In the reading phase, the extractors in the CPL subsystem extract both category and relation instances from unstructured free Web text using contextual patterns. Example of category and relation extraction patterns are “*mayor of X*” and “*X plays for Y*”, respectively. In the learning phase, such patterns are learned in CPL using some heuristic procedures-based co-occurrence statistics between noun phrases and existing contextual patterns (both defined using part-of-speech tag sequences) for each predicate of interest.
- **Coupled SEAL (CSEAL):** CSEAL is an extraction and learning system that extracts facts from semi-structured webpages using wrapper induction. Its core system is an existing wrapper induction system called SEAL (Wang and Cohen 2009).
- **Coupled Morphological Classifier (CMC):** CMC consists of a set of binary classifiers, one for each category, for classifying whether the extracted candidate facts/beliefs by other components/subsystems are indeed of their respective categories.
- **Rule Learner (RuleL):** RuleL is a first-order relational learning system similar to FOIL (Quinlan and Cameron-Jones 1993). Its goal is to learn probabilistic Horn clauses and use them to infer new relations from the existing relations in the KB.

Over the years, several new subsystem components were also proposed, e.g., Never Ending Image Learner (NEIL) (Chen et al. 2013), which classifies a noun phrase using its associated visual images, and OpenEval (Samadi et al. 2013), an online information validation technique, which uses real-time Web search to gather the distribution of text contexts around a noun phrase to extract instances of the predicates.

3.6 Summary

In this chapter, we have introduced methods for continuous and interactive learning of lexical and factual knowledge in dialogues. We have discussed various knowledge learning opportunities for chatbots during conversation with users. In regard to the scope for extracting factual knowledge from the dialogue context, we have discussed two systems namely, KNOWBOT (for learning knowledge graph from conversation) and KAD (for learning facts in a rule-based dialogue system). Next, we have presented approaches for lexical knowledge acquisition in dialogues. In particular, we have studied how chatbots can generate specific questions and form implicit requests to learn about unknown terms/concepts encountered during dialogue in the real-world. We have also presented two methods namely CILK and IKAI for interactive factual knowledge learning and inference in the dialogue-based question-answering setup. Finally, we have discussed NELL, a never-ending factual knowledge extraction system that chatbots can use to learn by extracting factual knowledge from the Web. This can be regarded as a passive approach of factual knowledge learning in conversation. Clearly, the current techniques are still in their infancy. There is still a lot to be done. We believe that this topic is not only interesting to dialogue systems researchers but also to human-computer interaction researchers. Their collaboration is more likely to produce more satisfactory dialogue systems.

References

- D. Ameixa, L. Coheur, P. Fialho, P. Quaresma, Luke, i am your father: dealing with out-of-domain requests by using movies subtitles, in *Intelligent Virtual Agents: 14th International Conference, IVA 2014, Boston, MA, USA, August 27–29, 2014. Proceedings 14* (Springer, 2014), pp. 13–21
- G. Angeli, M.J. Premkumar, C.D. Manning, Leveraging linguistic structure for open domain information extraction, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (2015), pp. 344–354
- R.E. Banchs, H. Li, Iris: a chat-oriented dialogue system based on the vector space model, in *Proceedings of the ACL 2012 System Demonstrations* (2012), pp. 37–42
- M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead, O. Etzioni, Open information extraction from the web, in *International Joint Conference on Artificial Intelligence* (2007)
- A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in *Advances in Neural Information Processing Systems*, 26 (2013)

- A. Bordes, J. Weston, R. Collobert, Y. Bengio, Learning structured embeddings of knowledge bases. *Proc. AAAI Conf. Artif. Intell.* **25**, 301–306 (2011)
- A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, T. Mitchell, Toward an architecture for never-ending language learning. *Proc. AAAI Conf. Artif. Intell.* **24**, 1306–1313 (2010)
- C.-H. Chang, M. Kaye, M.R. Girgis, K.F. Shaalan, A survey of web information extraction systems. *IEEE Trans. Knowl. Data Eng.* **18**(10), 1411–1428 (2006)
- Z. Chen, B. Liu, *Lifelong Machine Learning* (Morgan and Claypool Publishers, 2018)
- X. Chen, A. Shrivastava, A. Gupta, Neil: extracting visual knowledge from web data, in *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 1409–1416
- P. Clark, N. Balasubramanian, S. Bhakthavatsalam, K. Humphreys, J. Kinkead, A. Sabharwal, O. Tafford, Automatic construction of inference-supporting knowledge bases, in *4th Workshop on Automated Knowledge Base Construction (AKBC)* (Citeseer, 2014)
- M. Eric, L. Krishnan, F. Charette, C.D. Manning, Key-value retrieval networks for task-oriented dialogue, in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue* (2017), pp. 37–49
- A. Fader, S. Soderland, O. Etzioni, Identifying relations for open information extraction, in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (2011), pp. 1535–1545
- M. Gardner, P. Talukdar, J. Krishnamurthy, T. Mitchell, Incorporating vector space similarity in random walk inference over knowledge bases, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), pp. 397–406
- M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W.-T. Yih, M. Galley, A knowledge-grounded neural conversation model, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018)
- B. Hixon, P. Clark, H. Hajishirzi, Learning knowledge graphs for question answering through conversational dialog, in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2015), pp. 851–861
- J. Hoffart, F.M. Suchanek, K. Berberich, G. Weikum, Yago2: a spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.* 28–61 (2013)
- T. Kollar, V. Perera, D. Nardi, M. Veloso, Learning environmental knowledge from task-based human-robot dialog, in *2013 IEEE International Conference on Robotics and Automation* (IEEE, 2013), pp. 4304–4309
- N. Lao, E. Minkov, W. Cohen, Learning relational features with backward random walks, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (2015), pp. 666–675
- N. Lao, T. Mitchell, W. Cohen, Random walk inference and learning in a large scale knowledge base, in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (2011), pp. 529–539
- N. Lao, W.W. Cohen, Relational retrieval using a combination of path-constrained random walks. *Mach. Learn.* **81**, 53–67 (2010)
- P. Le, M. Dymetman, J.-M. Renders, Lstm-based mixture-of-experts for knowledge-aware dialogues, in *Proceedings of the 1st Workshop on Representation Learning for NLP* (2016), pp. 94–99
- J. Li, W. Monroe, D. Jurafsky, Data distillation for controlling specificity in dialogue generation (2017a). [arXiv:1702.06703](https://arxiv.org/abs/1702.06703)
- J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, D. Jurafsky, Adversarial learning for neural dialogue generation, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017b), pp. 2157–2169
- B. Liu, C. Mei, Lifelong knowledge learning in rule-based dialogue systems (2020). [arXiv:2011.09811](https://arxiv.org/abs/2011.09811)

- Y. Long, J. Wang, Z. Xu, Z. Wang, B. Wang, Z. Wang, A knowledge enhanced generative conversational service agent, in *Proceedings of the 6th Dialog System Technology Challenges (DSTC6) Workshop* (2017)
- R. Lowe, N. Pow, I. Vlad Serban, J. Pineau, The ubuntu dialogue corpus: a large dataset for research in unstructured multi-turn dialogue systems, in *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2015), pp. 285–294
- A. Madotto, C.-S. Wu, 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* (2018), pp. 1468–1478
- S. Mazumder, B. Liu, Context-aware path ranking for knowledge base completion, in *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (AAAI Press, 2017), pp. 1195–1201
- S. Mazumder, B. Liu, N. Ma, S. Wang, Continuous and interactive factual knowledge learning in verification dialogues, in *NeurIPS-2020 Workshop on Human and Machine in-the-Loop Evaluation and Learning Strategies* (2020)
- S. Mazumder, B. Liu, S. Wang, N. Ma, Lifelong and interactive learning of factual knowledge in dialogues, in *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue* (2019), pp. 21–31
- A. Mirza, M. Nagori, V. Kshirsagar, Constructing knowledge graph by extracting correlations from wikipedia corpus for optimizing web information retrieval, in *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (IEEE, 2018), pp. 1–7
- T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, Never-ending learning, in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-15)* (2015)
- T. Nayak, N. Majumder, P. Goyal, S. Poria, Deep neural approaches to relation triplets extraction: a comprehensive survey. *Cogn. Comput.* **13**, 1215–1232 (2021)
- A. Neelakantan, B. Roth, A. McCallum, Compositional vector space models for knowledge base completion, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (2015), pp. 156–166
- M. Nickel, K. Murphy, V. Tresp, E. Gabrilovich, A review of relational machine learning for knowledge graphs. *Proc. IEEE* **104**(1), 11–33 (2015)
- C. Niklaus, M. Cetto, A. Freitas, S. Handschuh, A survey on open information extraction, in *Proceedings of the 27th International Conference on Computational Linguistics* (2018), pp. 3866–3878
- K. Ono, R. Takeda, E. Nichols, M. Nakano, K. Komatani, Lexical acquisition through implicit confirmations over multiple dialogues, in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*(2017), pp. 50–59
- K. Ono, R. Takeda, E. Nichols, M. Nakano, K. Komatani, Toward lexical acquisition during dialogues through implicit confirmation for closed-domain chatbots, in *Proceedings of Second Workshop on Chatbots and Conversational Agent Technologies (WOCHAT)* (2016)
- T. Otsuka, K. Komatani, S. Sato, M. Nakano, Generating more specific questions for acquiring attributes of unknown concepts from users, in *Proceedings of the SIGDIAL 2013 Conference* (2013), pp. 70–77
- A. Padmakumar, P. Stone, R. Mooney, Learning a policy for opportunistic active learning, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (2018), pp. 1347–1357

- J.R. Quinlan, R.M. Cameron-Jones, Foil: a midterm report, in *European Conference on Machine Learning* (Springer, 1993), pp. 1–20
- M. Samadi, M. Veloso, M. Blum, Openeval: Web information query evaluation. Proc. AAAI Conf. Artif. Intell. **27**, 1163–1169 (2013)
- I.V. Serban, R. Lowe, P. Henderson, L. Charlin, J. Pineau, A survey of available corpora for building data-driven dialogue systems: the journal version. *Dialogue & Discourse* **9**(1), 1–49 (2018)
- J. Thomason, A. Padmakumar, J. Sinapov, J. Hart, P. Stone, R.J. Mooney, Opportunistic active learning for grounding natural language descriptions, in *Conference on Robot Learning* (PMLR, 2017), pp. 67–76
- O. Vinyals, Q. Le, A neural conversational mode (2015). [arXiv:1506.05869](https://arxiv.org/abs/1506.05869)
- R.C. Wang, W. Cohen, Character-level analysis of semi-structured documents for set expansion, in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing* (2009), pp. 1503–1512
- R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, D. Lin, Knowledge base completion via search-based question answering, in *Proceedings of the 23rd International Conference on World Wide Web* (ACM, 2014), pp. 515–526
- C. Wise, M.R. Calvo, P. Bhatia, V. Ioannidis, G. Karypus, G. Price, X. Song, R. Brand, N. Kulkani, Covid-19 knowledge graph: accelerating information retrieval and discovery for scientific literature, in *Proceedings of Knowledgeable NLP: the First Workshop on Integrating Structured Knowledge and Neural Networks for NLP* (2020), pp. 1–10
- F. Wu, D.S. Weld, Open information extraction using wikipedia, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (2010), pp. 118–127
- C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, W.-Y. Ma, Topic aware neural response generation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31 (2017)
- B. Yang, W.-T. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, in *International Conference on Learning Representations* (2015)
- T. Young, E. Cambria, I. Chaturvedi, H. Zhou, S. Biswas, M. Huang, Augmenting end-to-end dialogue systems with commonsense knowledge, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018)
- S. Zhang, K. Balog, Web table extraction, retrieval, and augmentation: a survey. *ACM Trans. Intell. Syst. Technol. (TIST)* **11**(2), 1–35 (2020)
- H. Zhou, T. Young, M. Huang, H. Zhao, J. Xu, X. Zhu, Commonsense knowledge aware conversation generation with graph attention, in *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (AAAI Press, 2018), pp. 4623–4629



Continuous and Interactive Language Learning and Grounding

4

Many task-oriented chatbots and virtual assistants like *Siri*, *Alexa*, and *Google Assistant* are built as Natural Language (command) Interfaces (NLI) that allow users to issue natural language (NL) commands to be mapped to some actions for execution in the underlying application in order to accomplish some tasks intended by the users. A fundamental feature of such systems is the ability to understand users' language and ground them to intended actions (often in symbolic form). Due to their diverse and wide-spread real-world applications, such NLI systems have driven research in language understanding, grounding and human-robot interactions over the years. Some of the prominent topics and applications include *robot navigation* (Artzi and Zettlemoyer 2013; Fried et al. 2018a; Guu et al. 2017; Tellex et al. 2011; Thomason et al. 2015) that focuses on grounding NL navigation instructions; *database querying* (Baik et al. 2019; Ferré 2017; Li et al. 2019; Li and Rafiei 2018; Liang 2016; Neelakantan et al. 2017; Xiong and Sun 2019; Zelle and Mooney 1996; Zettlemoyer and Collins 2007) that maps NL questions into executable queries (e.g., mapping text to SQL for relational databases and supporting question-answering over knowledge bases (Berant et al. 2013; Yih et al. 2015)); *visual data analysis* (Gao et al. 2015; Lawrence and Riezler 2016; Setlur et al. 2016; Utama et al. 2018) focusing on building NLIs for interactive data exploration; building NLI for invoking web APIs and interacting with GUIs (Branavan et al. 2010; Mazumder and Riva 2021; Pasupat and Liang 2015; Pasupat et al. 2018; Soh 2017; Su et al. 2017); NLI for Bash commands (Lin et al. 2018) and so on.

Existing methods for building such NLIs can be grouped into two broad categories. The first category views the process as *end-to-end modeling*, where a NL command is provided and the system directly outputs the action to be performed. For example, the authors of (Branavan et al. 2009; Fried et al. 2018b; MacMahon et al. 2006; Misra et al. 2017; Tellex et al. 2020; Vogel and Jurafsky 2010) have explored deep learning and reinforcement

learning to ground NL commands directly into executable actions. The other category focuses on learning a *semantic parser* (SP) to parse the NL command from the user into an intermediate logical form and then, translate the logical form into an executable action in the application (Andreas and Klein 2015; Artzi and Zettlemoyer 2013; Li and Rafiei 2018; Zelle and Mooney 1996; Zettlemoyer and Collins 2005). Despite their successes in building NLI, these approaches suffer from some key limitations:

- (1) They are trained with a large amount of application-specific training data which is often very hard to come by. Those deployed real-life systems that are written with handcrafted rules also need a huge amount of effort on the part of the human engineers. This leads to *high bootstrapping cost*.
- (2) For a different application, the model needs to be retrained using an application specific corpus. If a SP approach is taken, for each application, a different translator is needed too. This leads to *high fine-tuning cost*.
- (3) These systems do not have the ability to continually learn by themselves in a lifelong manner. Note, no matter how large a training dataset is collected and how many rules are written, it's hard to cover all possible language variations that users can come up with.

To deal with these challenges, it is important that such NLI systems acquire the ability to *learn continually through interactions with end-users* and *self-adapt* across applications/domains (Liu and Mazumder 2021; Mazumder 2021a). User interaction can be leveraged to learn new language expressions and ground them to suitable actions during conversation which can greatly improve the performance of NLI systems (Mazumder et al. 2020b).

Interactive learning also provides scope for improving scalability and robustness as the system can learn new actions from users through interactions as well. The bootstrapping cost could be significantly reduced because an interactive system does not need to be highly accurate to be deployed. On the other hand, such interactions enable explainability of the AI model by helping users to understand the reasoning behind the system's interpretation (Su et al. 2018) besides the scope for domain adaptation and personalization. A related research area on this regard is *Interactive Task Learning* (ITL) (Gluck and Laird 2018), broadly defined as “*any process by which an agent (A) improves its performance (P) on some task (T) through experience (E), where E consists of a series of sensing, effecting, and communication interactions between A, its world, and crucially other agents in the world.*” Chai et al. (2018) have provided a brief introduction to ITL where humans can teach physical agents new tasks through natural language communication and action demonstration. They discuss a specific form of ITL-*communicative task learning*, where humans can teach embodied agents (e.g., robots) in a shared physical world through language communication and action demonstration.

In this chapter, we discuss the scopes for continual and interactive language learning in the context of NLI and introduce some of the representative works along this direction. In the following sections, we discuss the modes of language learning in human-chatbot interac-

tions (Sect. 4.1), interactive language learning in the language game setting and through demonstration (Sect. 4.2); building self-adaptive NLI systems using dialogue-based knowledge acquisition and learning (Sect. 4.3), interactive semantic parsing and learning from user feedback for database querying applications (Sect. 4.4) and finally, conclude with a chapter summary (Sect. 4.5).

4.1 Modes of Language Learning in Human-Chatbot Interactions

Based on the various modalities of human-chatbot interactions, we organize the scope for learning new language expressions in the following two categories:

- **Learning via user demonstrations.** In some cases, NLI systems deployed in practice come with *Graphical User Interfaces* (GUIs) or remote control facilities to explicitly control devices apart from controlling them via NL commands. Examples of such systems include task completion robots performing household activities like cleaning and personal assistant services integrated with home appliances such as Smart TVs, Smart Lights, Smart Speakers, etc. Considering the user has issued an NL command and the bot has failed to execute the intended action, the user may perform the intended action via the GUI or remote control. The bot can then record the sequence of executable action(s) performed by the user by accessing the underlying application logs and store the executed APIs as the ground truth for the original input NL command. The command along with the invoked APIs can serve as labeled examples for learning the command. Some of the related research include learning language through interactive demonstrations (Wang et al. 2016, 2017) and teaching robots new tasks through demonstrations and instructions (Argall et al. 2009; Mohseni-Kabir et al. 2019; Rybski et al. 2007; Thomaz and Cakmak 2009).
- **Learning via multi-turn NL dialogues with the user.** In many cases, demonstration may not be possible either because the user does not know how to do it or no GUI or remote control exists. Learning language via multi-turn NL dialogues with the end-user may be the only option. For example, given the user has issued the command “*turn off the light in the kitchen*” and the bot has failed to understand or ground the command, it can show/tell a list of *top-k* predicted actions as NL descriptions (as shown below) that can be executed in the current state:

Bot: Sorry, I did not get you. Do you mean

option-1. switch off the light in the kitchen, or

option-2. change the color of the light in the kitchen, or

option-3. switch on the light in the kitchen?

The user can easily select the right option (option-3). The action API [say, `SwitchOffLight(arg:place)`] corresponding to the selected option (here, option-3) can be regarded as the ground truth for the issued NL command, which is to be used as a new example for learning the new language expression. In subsequent turns of the dialogue session, the agent can also ask additional NL questions and show option list to acquire the ground truth values of the arguments of the (ground truth) action API. Recently, Mazumder et al. (2020b, 2021a) have proposed a method to enable dialogue based language learning in the context of building NLI. Other related works include (Ping et al. 2020; Tolkahev et al. 2022). We discuss more about these works in Sect. 4.3. In the robotics domain, some of the works have also explored the use of natural language and dialogue to teach robots new actions (Mohan and Laird 2014; Scheutz et al. 2017) and there are also approaches that have used *both* natural language communication and action demonstration to teach robots new tasks (Liu et al. 2016; She and Chai 2016, 2017; She et al. 2014).

4.2 Learning Language through Interactions and Demonstration

Acquisition of generic knowledge in Humans is often facilitated through linguistic communication and manual demonstration, formally termed as *natural pedagogy* (Csibra and Gergely 2006). Through childhood to adulthood, humans have developed the ability to learn and teach through natural pedagogy, which appears to be universal across cultures and can be traced back to our ancestors (Tehrani and Riede 2008). This idea of learning through natural pedagogy has driven many research contributions in the field of interactive language learning and grounding over the years. In this section, we discuss some of the representative ones. Although, many of these works and their proposed models do not directly follow the traditional Lifelong/Continual Learning (CL) setting as discussed in Chap. 2, we think that they are very relevant for the CL research in this domain.

4.2.1 Learning Language Games through Interaction

The semantics of language is derived from its usage. Motivated by this idea, Wittgenstein (2010) has introduced the idea of *language games* and shown how a builder B and an assistant A can use a primitive language consisting of four words, viz., ‘*block*’, ‘*pillar*’, ‘*slab*’, ‘*beam*’ to successfully communicate between themselves for passing blocks from A to B. Recently, Suglia et al. (2020), Wang et al. (2016), Yu et al. (2017) have studied this concept of language games in interactive learning setting, where they showed how an AI agent can learn language of the user through feedback and demonstrations while cooperatively accomplishing a task.

In doing so, the agent gradually adapts itself by learning new language expressions from the user and improves the task completion performance over time.

Formally, the idea of language games is described as follows,

- A human (user) wishes to accomplish a task (e.g., achieving a certain configuration of blocks), but can only communicate with a computer who performs the actual actions (e.g., removing all red blocks).
- The computer initially knows nothing about the language and therefore must learn it from scratch through interactions, while the human adapts to the computer’s capabilities.

Some of the related research in this area are as follows. Yu et al. (2017) have created a dataset for such interactive learning. Suglia et al. (2020) have introduced an “imagination” module that learns context-aware and category-aware latent embeddings without relying on category labels at inference time, which is usually required by prior works. De Vries et al. (2017) have introduced a two-player guessing game as a testbed for research on the interplay of computer vision and dialogue systems, called *GuessWhat?!* The goal of the game is to locate an unknown object in a rich image scene by asking a sequence of questions. Shekhar et al. (2019) have proposed a grounded dialogue state encoder that can integrate visual grounding with dialogue system components. Below, we describe a representative related work—*Interactive learning through language games* (ILLG) (Wang et al. 2016) to introduce this area of research.

Wang et al. (2016) have defined the ILLG setting where the objective is to transform a start state into a goal state, but the only action the human can take is entering an utterance. To accomplish the goal, the computer parses the utterance and produces a ranked list of possible interpretations based on its current model. Human scrolls through the list and chooses the intended one, simultaneously advancing the state of the blocks and providing feedback to the computer. For the computer to be successful, it has to learn the human’s language quickly over the course of the game so that the human can accomplish the goal more efficiently (i.e., with as little scroll as possible).

To study the problem of ILLG, Wang et al. (2016) have designed a game called SHRD-LURN (see Fig. 4.1), in homage to the seminal work of (Winograd 1972). We describe the game below.

The SHRD-LURN Game. The game consists of two players—the *human* and the *computer* and proceeds through a fixed number of levels. In each level, the game starts with a starting state $s \in \mathcal{Y}$ presented to both players and a goal state $t \in \mathcal{Y}$ which is only visible to the human. Here, \mathcal{Y} is the set of all configurations of blocks. In the game designed by Wang et al. (2016) and as shown in Fig. 4.1, each state consists of stacks of colored blocks arranged in a line, where each stack is a vertical column of blocks.

Given the initial state s (block configuration), the human provides an utterance x (e.g., “remove red”) to the computer which then produces a ranked list of candidate actions $Z = [z_1, \dots, z_K] \subseteq \mathbb{Z}$ (e.g., `remove(with(red))`, `add(with(orange))` etc.), where \mathbb{Z}

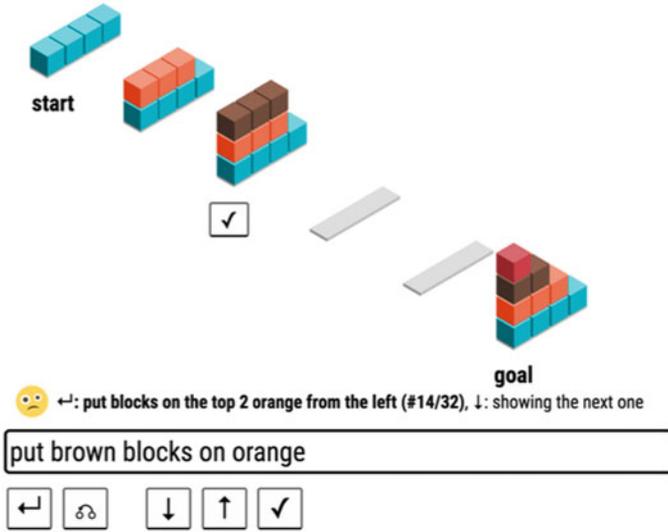


Fig. 4.1 The SHRDLURN game (Source (Wang et al. 2016))

Table 4.1 The formal grammar defining the compositional action space \mathbb{Z} for SHRDLURN. Here, c is used to denote a Color, and s to denote a Set (Source (Wang et al. 2016))

Rule	Semantics	Description
Set	all()	All stacks
Color	cyan violet red orange	Primitive color
Color \rightarrow Set	with(c)	Stacks whose top block has color c
Set \rightarrow Set	not(s)	All stacks except those in s
Set \rightarrow Set	leftmost rightmost(s)	Leftmost/rightmost stack in s
Set Color \rightarrow Act	add(s, c)	Add block with color c on each stack in s
Set \rightarrow Act	remove(s)	Remove the topmost block of each stack in s

is the set of all possible actions. The actions here are defined compositionally via the grammar in Table 4.1. Each action either adds to or removes from a set of stacks and a set of stacks is computed via various set operations and selecting by color. For example, the action $\text{remove}(\text{leftmost}(\text{with}(\text{red})))$ removes the top block from the leftmost stack whose top most block is red.

For each $z_i \in \mathbb{Z}$, the computer then computes the successor state $y_i = \text{exec}(z_i, s)$ obtained due to execution of z_i on state s and returns the ordered list $Y = [y_1, \dots, y_k]$ of successor states of s to the human. The human then chooses y_i from the list Y to indicate

y_i to be the correct action for utterance x . The game state is then updated to $s = y_i$. Note, here if the human chosen y_i where $i = 1$, then the computer’s prediction is considered to be *correct* for x . The level ends when $s = t$, i.e, the level reaches the goal state and then, the next level starts.

Given this setting, the game is expected to proceed as follows. Initially, the computer does not understand human’s utterance and performs arbitrary actions. But as the computer obtains feedback from the human and learns the language, the two should become more proficient at communicating and playing the game. Thus, if the computer can learn the language of the human well, it should achieve good game performance.

A Semantic Parsing Model for ILLG. Wang et al. (2016) have built a semantic parsing model to solve the ILLG problem, where the goal of the semantic parser is to map a NL utterance (e.g., ‘*remove red*’) from human into logical forms (e.g., `remove(with(red))`). Given an utterance x , a log-linear model is trained over logical forms (actions) $z \in \mathbb{Z}$ to model the probability of an action being the correct interpretation, as:

$$p_\theta(z|x) \propto \exp(\theta^\top \phi(x, z)), \quad (4.1)$$

where $\phi(x, z) \in \mathbb{R}^d$ is a feature vector and $\theta \in \mathbb{R}^d$ is parameter of the model.

The features used are n -grams (including skip-grams) in the utterance **conjoined** with tree-grams on the logical form side. Given an utterance (e.g., ‘*stack red on orange*’), uni-grams (*stack*, *, *), bigrams (*red*, *on*, *), trigrams (*red*, *on*, *orange*), and skip-trigrams (*stack*, *, *on*) are extracted. For a candidate logical form, the features involve the predicates in the logical forms and their arguments. For each predicate h , let $h.i$ be the i -th argument of h . Then, the *tree-gram* features $\psi(h, d)$ for predicate h and depth $d = 0, 1, 2, 3$ are:

$$\begin{aligned} \psi(h, 0) &= \{h\}, \\ \psi(h, d) &= \{(h, i, \psi(h.i, d - 1)) \mid i = 1, 2, 3\}. \end{aligned}$$

The set of all features is just the cross product of utterance features and logical form features.

The parser generates logical forms over \mathbb{Z} from the smallest to the largest using beam search. In this process, for a given size n , a set of logical forms of size n (with exactly n predicates) is constructed by combining logical forms of smaller sizes according to the grammar rules in Table 4.1 and logical forms with the highest score $\theta^\top \phi(x, z)$ are retained according to the current model θ . Let Z be the set of logical forms on the final beam, consisting of logical forms of all sizes. The system executes all of the logical forms on the final beam Z , and orders the resulting denotations y by the maximum probability of any logical form that produced it.

Learning. When the human provides feedback in the form of a particular y , the following loss function is formed:

$$\ell(\theta, x, y) = -\log p_{\theta}(y | x, s) + \lambda \|\theta\|_1, \quad (4.2)$$

$$p_{\theta}(y|x, s) = \sum_{z:exec(z,s)=y} p_{\theta}(z|x). \quad (4.3)$$

and then, it makes a single gradient update using AdaGrad (Duchi et al. 2011).

4.2.2 Language Learning through Collaborative Instruction Following

Many works have studied the problem of language learning in the game setting, where the goal is to develop interactive agents that can communicate with humans to collaboratively solve tasks in grounded scenarios. The task is formulated as an instruction following problem, where the user (human) provides a NL instruction to follow for the agent and the agent aims to interpret, ground and execute it correctly to proceed in the game. The interaction is collaborative in the sense that human and agent cooperate to complete a task jointly. We briefly discuss some of the related works here.

Narayan-Chen et al. (2019) have introduced the Minecraft Collaborative Building Task, in which pairs of users control avatars in the Minecraft virtual environment and collaboratively build 3D structures in a Blocks World-like scenario while communicating solely via text chat. In particular, one player (A, the Architect) is shown a target structure, who needs to instruct the other player (B, the Builder) to build this structure. Both players interact via a chat interface. A can observe B but cannot place blocks. They have built a data collection platform and have used it to collect the *Minecraft Dialogue Corpus*, consisting of 509 human-human written dialogues, screenshots and complete game logs for this task. The goal of the work is to develop fully interactive agents that can collaborate with humans successfully on this task.

Bisk et al. (2018) have studied the problem of mapping NL instructions to complex spatial actions in a 3D blocks world. They have introduced a dataset that pairs complex 3D spatial operations to rich NL descriptions that require complex spatial and pragmatic interpretations such as “*mirroring*”, “*twisting*” and “*balancing*”. An end-to-end trainable model is also proposed that takes in a NL instruction for block manipulation and a 3D representation of the world as input, and outputs where the chosen block should be moved to (by predicting a coordinate in space).

McCarthy et al. (2021) have investigated how humans use natural language to collaboratively solve physical assembly problems more effectively over time. They have paired up Human participants in an online environment to reconstruct scenes containing two block towers. One participant is able to see the target tower and sends assembly instructions for the other participant to reconstruct. Participants have provided increasingly concise instructions across repeated attempts on each pair of towers, using more abstract referring expressions that capture each scene’s hierarchical structure.

Gray et al. (2019) have developed a framework for dialogue-enabled interactive agents in Minecraft CraftAssist to facilitate the study of agents that can complete tasks specified by dialogue and eventually, to learn from dialogue interactions. Srinet et al. (2020) have proposed a semantic parsing dataset focused on instruction-driven communication with an agent in Minecraft. The dataset consists of 7K human utterances and their corresponding parses. Given a proper world state, the parses can be interpreted and executed in game.

Effenberger et al. (2022) have provided an analysis of language change in collaborative instruction following. They have found that utility-maximizing participants trade surface form linguistic complexity with established norms, as the familiarity and expertise of the interaction partners increase. In practice, this emerges as a reduction in utterance length and vocabulary size.

As we have discussed before, *semantic parsing* and *end-to-end modeling* are the two of the most common approaches to language grounding. *Semantic parsers* (SP) are often built on top of fixed object categories. Thus, they are precise but inflexible. On the other hand *end-to-end models* are maximally expressive, but fickle and opaque. Burns et al. (2021) have developed a system that balances the strengths of these approaches so that users can teach agents new instructions that generalize broadly from a single example. They introduce the idea of *neural abstractions*, which defines a set of constraints on generative models to enable users to build a SP that can generalize across contexts. Starting from a core programming language that operates over abstractions, users can define increasingly complex mappings from natural language to actions. The system is evaluated in Minecraft using the CraftAssist (Gray et al. 2019) framework. Here, users are placed in a Minecraft session with a language agent and are instructed to make any desired modifications to a given house. However, they can only talk to the agent for achieving the goal. The work has shown that over time, users rely increasingly on induced utterances and the number of failed parses decreases dramatically.

Rubavicius and Lascarides (2022) have studied the problem of symbol grounding in a visual reference resolution task—the learner leverages an embodied natural language (NL) utterance from the teacher and its context and learn to map NL *referential expressions* (REs) [e.g. “*all but one green square above both red circles*”] in the utterance to their denotations, given the visual percepts. They introduce a procedure that combines (1) lazy few shot learning to relate open-class words like “*green*” and “*above*” to their visual percepts and (2) symbolic reasoning with closed-class word categories like *quantifiers* and *negation*. The combination is used to estimate new training examples for grounding symbols that occur within a noun phrase but are not designated by that noun phrase [e.g. *red* (in noun phrase “*red circles*”) in the above example] and gain data efficiency.

4.3 Dialogue-Driven Learning for Adaptive NLI

The interactive language learning approach discussed in Sect. 4.2 is suitable for situations where users can provide demonstrations and have knowledge about how to perform the task. Another approach (as discussed in Sect. 4.1) is to learn to ground language via multi-turn NL dialogue with the users. Below, we discuss some of the representative works that aim to learn new concepts and/or language patterns through dialogue for building adaptive NLI systems.

4.3.1 Interactive Teaching of New Concepts for Adaptive NLI

Most Conversational AI systems rely on knowledge-based or machine-learning based (or hybrid) understanding components under the hood to understand the intents of the users and their arguments (usually called *slots*), such as “*set an alarm for 7 am*”, where the intent can be *SET_ALARM*, and *7 am* can be the *TIME* slot. These systems typically do not have the capability to interact with end-users during a live conversation to seek explanations to improve the AI system’s understanding of entities, intents or other constructs. When users speak an utterance outside the comprehension of these systems, the AI agent responds with a dead-end response such as “*Sorry I don’t know that*”.

Motivated by how children learn their first language interacting with adults, Ping et al. (2020) have described a *Teachable AI Dialogue system* that is capable of learning new language nuggets called *concepts*, directly from end users using live interactive teaching sessions using natural language explanations. The system augments goal-oriented AI agents with an interactive teaching capability, to improve task completion rates by asking questions to fill gaps in the AI agent’s understanding, hence making the agent more personal, conversational and adaptive.

Figure 4.2 shows the architecture of the teachable dialogue system that acts as a subsidiary to the multi-domain Conversational AI system and springs into action when the AI agent is not able to successfully interpret/execute a user’s input utterance. This way, the parent Conversational AI system remains decoupled from the teachable dialogue system, staying independent of user-specific interpretation or modeling.

Components of the Teachable Dialogue System. The system consists of three components: (1) a multi-task neural *Concept Parser* to identify gaps in automatic understanding during live conversational interactions. In particular, the agent checks the utterances whether it is “Teachable” or not and identifies the *entity or intent* which needs to be learned from the users. If a gap is identified, the “Classroom” sub-dialogue kicks in, where the users are asked what they mean by these concepts. (2) *Definition Understanding model* that extracts and interprets explanations from the user and maps the unknown concept phrases to the learned definitions. (3) *Dialog Policy model* that drives teaching conversations with a goal-completion objective

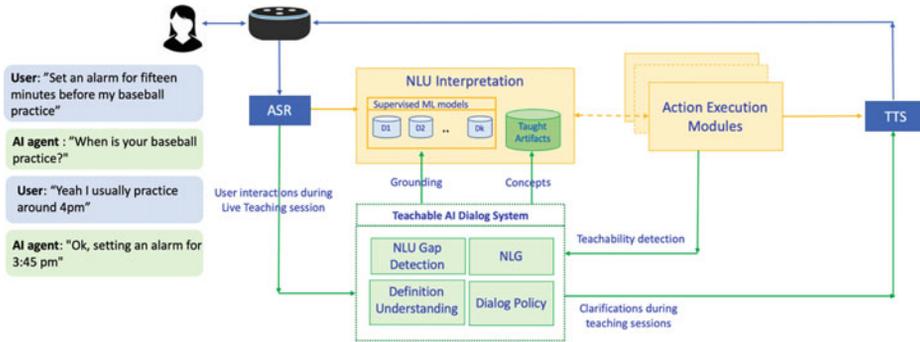


Fig. 4.2 Conversational AI system using a Teachable Dialog system (Source (Ping et al. 2020))

of extracting necessary explanations from the user and ground the respective unknown concepts. These components interact with each other to facilitate concept learning as follows.

Concept Parser identifies the sequence of tokens (called *concept phrases*) that are not understood by the parent NLU system and helps in initiating a teaching session with the user by asking the question—“*Can you teach me what you mean by <concept phrase> ?*” Once a teaching session has been initiated with the user by the dialogue system, the Dialog Policy model helps in predicting the right clarification question to ask the user, based on the context of the original utterance and the interactions with the user during the teaching session. The answers from the user are then processed by the Definition Understanding model that works along with the policy models to conduct multi-turn conversations with the user to learn, clarify and ground the respective concepts in the user’s utterance. When the teachable dialogue system deems a teaching session as successful, the taught actions are subsequently executed by the AI system, and the respective definitions and artifacts are then stored for future re-use.

Concept Parser. Concept parser accurately localizes the segments within an utterance that an NLU system is unable to comprehend and needed to execute the relevant action. In addition to using a slot-tagger objective in the Concept Parser model, Ping et al. (2020) have also added an auxiliary task of semantic-chunking to make sure the model is sensitive to segments that might be neglected by a token-level slot tagger. For example, the chunking model should be able to segment the sentence: “[show] [me] [navigation] [to] [where we go camping every year]” while the slot tagger may not correctly tag “where we go camping every year” as “Location” concept. The parser is implemented using a multi-task transformer model which is end-to-end trainable using slot tagging, semantic chunking and concept relevance task objectives.

Definition Understanding. Once Concept Parser has detected a teachable phrase, a teaching session is initiated by requesting the user to provide a suitable definition of the identified concept. The Definition Understanding module attempts to accurately extract the appropriate definition from the user’s answer, which can be verbose (indirect answer),

distracted (do not intend to answer the definition question), incomplete (not sufficient enough to fully ground), complicated (difficult to understand by the parent AI system) or have contextual references such as “*make it brighter*”. The module uses a Transformer-based joint intent and span detection framework that takes in the dialogue history, user answer and the slot type and performs intent and span classification of the user answer to solve the definition extraction task.

Dialogue Policy. The Dialogue Policy module predicts the next action in a multi-turn teaching dialogue session with a goal-completion target of extracting all the necessary definitions from the user during teaching. A successful completion of the teaching session is one where the concept phrase in the first-turn utterance has been grounded to an equivalent action that is executable by the parent AI system. The Dialogue Policy uses a Transformer-based contextual model to predict the next action during a teaching session. The action-space for the model includes (1) Ask or repeat a clarification question, (2) Guardrail the conversations to channelize users back into the teaching conversations, (3) Identify Out-Of-Domain (OOD) turns during a teaching session, (4) Ground the extracted definitions with the parent NLU system, and (5) deem a teaching session as successful or unsuccessful, or decide to end a teaching session. The module takes in utterances from contextual dialogue history, predicted definition spans, confidence scores from the Definition Understanding module and the set of recognized / resolved slots from the parent NLU model and passes the input through a Transformer-based encoder to get the last layer output of [CLS] token as a contextual representation of the inputs. Then, a single feed-forward layer and a softmax layer is applied to obtain a probability distribution over all the possible actions.

4.3.2 Learning Adaptive NLI’s via Command Matching and Learning

Mazumder et al. (2020b, 2021a) have proposed a continuous learning system for NLI’s, called *Command Matching and Learning* (CML) with dialogue-driven language learning and grounding capabilities. CML is based on the SOLA framework in Chap. 2. In that chapter, we used CML as an example to illustrate the SOLA framework. This section provides more details. CML enables NLI’s to learn new paraphrased commands that it doesn’t understand via an interactive dialogue with the user. When the same or a similar command is issued in the future, the system will have no problem in understanding it.

The CML Approach. CML is based on the approach called *NL2NL* (natural language to natural language) matching to automatically build NLI’s for API (Application Programming Interface)-based applications such as robot navigation and action command systems, virtual assistants like issuing a service request to Siri and Alexa, and GUI-based software applications (e.g., manipulating objects in MS Paint). An API-based application has a finite set of APIs \mathcal{A} (each $a_i \in \mathcal{A}$ having a list of arguments) that can be invoked to perform some actions in the application. When an API a_i is grounded and invoked, its arguments and their

assigned values tell the application which application object(s) the action is to be performed on. For example, considering the *Microsoft Paint* tool as an application and an (example) API `drawCircle(X1, X2)` [where $X1$ and $X2$ are variables representing the arguments of the API], it can be invoked to draw a circle (an application object) having a given **color** (specified by argument $X1$) at a location (denoted by **coordinate** $X2$). A grounding of the API can be `drawCircle(X1 = "blue", X2 = (20, 40))`, which tells the application to draw a blue circle at coordinate (20, 40) in the 2-D application space.

Given an API-based application, the NL2NL approach works as follows. The developer of the application first writes a set of *seed commands* (SCs) in natural language (NL) to represent each API action supported by the application. SCs are just like paraphrased NL commands from end users to invoke the action API. The only difference is that the objects to be acted upon in each SC are replaced with variables, which are the arguments of the action API. When the end-user issues a NL command, the system simply matches the command with a SC of the correct action and in doing so, it also grounds the variables/arguments for the associated action API to be executed. For the aforementioned API example, let “draw a $Z1$ circle at $Z2$ ” be a SC for the `drawCircle(X1, X2)` API, where the variable mappings between the API and the SC is: $\{Z1 = X1, Z2 = X2\}$. Then, a user command “insert a blue circle at (20, 40)” can be matched to this SC, where the grounded API arguments are $X1 = \text{'blue'}$ and $X2 = (20, 40)$.

Since the SCs written by the developer are unlikely to cover all possible paraphrased expressions that a user may use to express the same command. If the system fails to match a user command C (which corresponds to the *novelty detection* step in the SOLA framework described in Chap. 2), it interacts with the user to gain the ground-truth action that the user has requested, which corresponds to the *adaptation* step in SOLA. To facilitate interactions with the user, the matching process also identifies likely SCs or API actions that the user has requested. This corresponds to the *novelty characterization* step of SOLA. After gaining the ground truth action, the system learns the command, which corresponds to the *continual learning* step of SOLA. That is, the system learns the new command from the end-user via a simple dialogue to create a new (paraphrased) SC and (paraphrased) argument values. The risk management in SOLA is also used. For example, CML does not ask the user too many questions in the interactive process in order to not annoy the user.

Components of CML. CML has three components: (1) an *application specification*, which enables the application developer to specify a set of SCs for each of their APIs, (2) a *command grounding module*, that grounds a user command C to an action SC by matching C with the correct SC (whose associated action API is then executed), and (3) an *interactive continual learner* that interacts with users to learn new SCs and paraphrases of API argument values.

Application Specification. CML assumes an application comes with two sets of APIs: *action APIs* \mathcal{A} and *utility APIs* \mathcal{U} . \mathcal{A} defines a set of actions that can be performed in the application. Each action $a_i \in \mathcal{A}$ causes a change in the *state* of the (instantiated) objects in the application, specified by objects’ properties and their values. For example, in *Microsoft*

APR(arg: arg type)	Seed Commands (SCs)	Example User Command
SwitchOnLight(X1: location)	1. Switch on the light in X1 2. Put on light in X1	Switch on the light in the bedroom (X1) .
SwitchOffLight(X1: location)	1. Switch off the light in X1 2. Put off light in X1	Switch off the light in the bedroom (X1)
ChangeLightColor (X1: location, X2: color)	1. Change the X1 light to X2 2. I want X1 light to be X2	Change the bedroom (X1) light to blue (X2)

Fig. 4.3 Three APIs, their SCs and some example commands

Paint, a circle drawn on the editor is an example of an *instantiated* object and it can have properties like *color*, *name*, *shape*, etc, with their values like the color *red*. Examples of actions are: *draw a circle*, *change the shape of the circle*.

Utility APIs in \mathcal{U} are used to resolve *referential expressions* in user commands. For example, in the user command: “*move the blue block to the left of the cube*”, “*blue block*”, “*cube*”, “*left of the cube*” are referential expressions, where “*blue block*” refers to some block having the color “*blue*”; “*cube*” refers to some block having the shape “*cube*” and “*left of the cube*” refers to some *location* that is on the left side of the cube shaped block and where the blue block is to be moved to. Utility APIs help in resolving referential expressions by identifying the referred objects (by object ids) having a given property value or a property value referred through an object in the current state of the application.

Application Specification consists of three parts: **(1) Properties and their domains**, **(2) Action SC specification**, and **(3) Utility SC specification**. Figure 4.3 shows three API actions (column 1) for switching on the light in a location, switching off the light in a location, and change the light color of a location in a SmartHome application. Two action SCs are given for each API action in column 2. An example user command (without X1 or X2) is given in column 3 for each API. Here, X1 and X2 are variables or place holders for location and color respectively. Additional details about action SC specification, utility SC specification, specification for properties and their domains, and an full example application can be found in (Mazumder et al. 2020b).

Command Grounding Module. Given a user command C and the application specification, the *command grounding module* (CGM) returns a grounded SC set, consisting of one *action* SC and a set of *utility* SCs, telling the system what action to perform. If the grounding is not successful, \emptyset is returned. CGM has two main sub-modules:

(i) *Rephraser and Tagger* (\mathcal{R}): Given the user command C , \mathcal{R} rephrases C and tags each word or phrase in the *rephrased* C with either ‘O’ (i.e., not an argument type) or one of the possible argument types of the action SCs to produce a tagged command C' . This work uses a dictionary lookup and regular expression-based rephraser and tagger for \mathcal{R} .

(ii) *SC Matcher [or simply Matcher]* (\mathcal{M}): Given the rephrased and tagged command C' and the set \mathcal{T} of (action or utility) SCs, *Matcher* \mathcal{M} computes a match score $f(t, C')$ for each $t \in \mathcal{T}$ and returns the top ranked SC $\hat{t} = \arg \max_{t \in \mathcal{T}} f(t, C')$. Any paraphrasing model can be used as \mathcal{M} . The work in (Mazumder et al. 2020b) uses information retrieval (IR) based unsupervised matching models for \mathcal{M} . Details can be found in the CML paper.

Interactive Continual Learner. As a given SC can have many paraphrased versions and the developer can only provide a limited number of them, SC Matcher may not find significant match for an unseen paraphrased user command (a novel command is detected). CML learns new SCs from users through interactions continually. Additionally, it also learns new paraphrases of argument (domain) values from users to improve \mathcal{R} over time (Mazumder et al. 2020b).

When the system detects a hard-to-understand user command, it tries to learn the command so that it will be able to understand it and similar commands in the future. The system assumes that every command is relevant to the application. The novelty characterization step of CML, which is also done by SC Matcher, identifies the part of the user command that the system does not understand and how similar it is to some known commands. Based on the characterization, the system adapts by asking the user via an interactive dialogue to obtain the ground truth API action requested by the user, which also serves as a piece of training data for continual learning. In the adaptation or accommodation process, risk is also considered.

Let us see an example based on the SC specifications in Fig. 4.3. The user issues the command “*turn off the light in the kitchen*” that the system does not understand (i.e., a novelty), i.e., the SC Matcher fails to ground/match the command with any of the SCs in Fig. 4.3. Based on the current system state, it decides which part of the command it can understand or ground, which part it has difficulty with, and what known commands are similar to the user command (i.e., characterization). Based on the characterization result, the interactive learner provides the user a list of top-k predicted actions (see below) described in natural language and asks the user to select the most appropriate action from the given list (i.e., adaptation).

User: Turn off the light in the kitchen

Bot: Sorry, I did not get you. Do you mean

option-1. switch off the light in the kitchen, or

option-2. switch on the light in the kitchen.

The user selects the desired action (option-1). The action API [SwitchOffLight(X1:location)] corresponding to the selected action (option-1) is retained as the ground truth action for the issued user command. In subsequent turns of the dialogue, the interactive learner will ask the user questions to acquire ground truth values associated with the arguments of the selected action, as defined in the API. This process is controlled by an action planner. CML then incrementally learns to map the original command “*turn off the light in the kitchen*” to the API action, SwitchOffLight(arg:location). Learning here means to create a new SC [turn off the light in X1] and add it to the list of SCs for the API SwitchOffLight(X1:location) so that in the future, when this user or any other user issues

the same or a similar command, CML will have no problem in understanding or grounding it, i.e., mapping the command to this new SC. Over time, CML learns more and more from the users and becomes more and more knowledgeable and powerful in serving them.

Risk is considered in CML in two ways. First, it does not ask the user too many questions in order not to annoy the user. Second, when the characterization is not confident, the system simply asks the user to say his/her command again in an alternative way (which may be easier for the system to ground or understand) rather than providing a list of random options for the user to choose from, which may make the user lose confidence in the system.

4.4 Interactive Semantic Parsing and Learning from Feedback

Given a NL utterance from the user, existing works on semantic parsing (Berant et al. 2013; Su et al. 2017; Yih et al. 2015; Yu et al. 2018; Zettlemoyer and Collins 2005) mostly aim to produce the intended logical form or action in *one shot*. In such approaches, even if the model miss-interprets the user's command and produces an incorrect action, there is no way to rectify the mistake on the fly and learn from it. Ideally, a semantic parser should be self-aware of its uncertainties (Dong et al. 2018) and prompt for user demonstrations when uncertain. Recently, a number of approaches have leveraged user interactions to improve semantic parsing. A majority of the approaches have focused on rectifying mistakes and resolving ambiguities of semantic parsers using user interaction and feedback. Continual learning from feedback through interactive dialogues in semantic parsing is still largely unexplored. In the following subsections, we first provide a brief summary of existing semantic parsing works that leverage user interactions and then, discuss a recent method that aims to learn semantic parser from user interaction in the continual learning setting.

4.4.1 Improving Semantic Parsing with User Interaction

There have been a number of works (Chaurasia and Mooney 2017; Elgohary et al. 2020; Gur et al. 2018; He et al. 2016; Li and Jagadish 2014; Semantic Machines et al. 2020; Su et al. 2018; Yao et al. 2019a, b) that have leveraged user interactions to improve semantic parsing on the fly. These works mainly attempt to detect potential errors in the generated SQL queries or ambiguities in parsing and ask the user clarification questions in order to resolve ambiguities and/or revise the generated SQL based on the user feedback. Artzi and Zettlemoyer (2013), Clarke et al. (2010), Iyer et al. (2017) have propose methods to learn semantic parsers from binary user feedback that indicates whether executing a generated query yields the correct results or not. Gur et al. (2018) have built a model to identify and correct error spans in the generated queries via dialogues. Yao et al. (2019b) have formalized a model-based intelligent agent MISP to enable user interaction via a policy probability-based uncertainty estimator, which is a grammar-based natural language

generator and a multi-choice question-answer interaction design. More recently, Elgohary et al. (2020) have crowdsourced a dataset for fixing incorrect SQL queries using free-form natural language feedback. Semantic Machines et al. (2020) have constructed a contextual semantic parsing dataset where agents could trigger conversations to handle exceptions such as ambiguous or incomplete user commands. In the program synthesis domain, Chaurasia and Mooney (2017) have proposed a system that engages users in a dialogue to clarify intents for generating computer code from NL descriptions. Yao et al. (2019b) have designed an interactive semantic parsing method, where the agent asks users clarification questions to resolve ambiguities for synthesizing “If-Then recipes” programs.

4.4.2 Learning Semantic Parser from User Interaction

A recent work (Yao et al. 2020) has proposed a human-in-the-loop methodology for learning semantic parsers in a continual manner. Instead of using user interaction to validate parsing results, they aim to learn semantic parsers from fine-grained interaction where users answer simple questions based on their background knowledge. In other words, the system proactively interacts with the user by asking simple questions to resolve the uncertainties regarding the parsing and in doing so, it gets to imitate the user behavior and continue improving itself autonomously in interpreting users’ questions.

Figure 4.4 shows an example, where the system is uncertain about which condition applies to the entity “*jalen rose*” in the question asked by the user and proactively ask the user questions to resolve ambiguities about appropriate table attributes for solving the query. Gradually, it produces a SQL query for the question from the user and on getting feedback from user, it learns the ground truth for the example, which is collected for continual learning and performance improvement.

One challenge in learning semantic parser through such fine-grained user interaction is that it may drive the system to ask too many questions and overwhelm the user. To make the approach feasible, the number of such fine-grained interactions needs to be limited. To tackle the problem, Yao et al. (2020) have developed an *a*n*n*otation-*E*fficient *I*mitation *L*earning algorithm called NEIL for learning semantic parsers from such sparse, fine-grained demonstrations. In NEIL, the agent (semantic parser) only requests for demonstrations when it is uncertain about a state (parsing step). For certain/confident states, actions chosen by the current policy are deemed correct and are executed to continue the parsing. The policy is updated iteratively in a Dataset Aggregation fashion (Ross et al. 2011). In each iteration, all the state-action pairs, demonstrated or confident, are included to form a new training set and train a new policy in a supervised manner. In the following subsections, we briefly discuss the problem formulation in NEIL and its imitation learning algorithm.

User Interaction

No.	Player	Nationality	School/Club Team	Position
25	Aleksandar Radojević	Serbia	Barton CC (KS)	Center
5	Jalen Rose	United States	Michigan	Guard-Forward
...

How many schools or teams had jalen rose?

What condition does "jalen rose" imply?
(System Uncertainty)

Does the system need to consider any conditions about the table attribute "School/Club Team"?

No.

I'm confused 😞 Please help me out! Should I consider conditions about any of the following table attributes? (1) "Player" (2) "Nationality" (3) "Position" (4) None of the above options.

(1) "Player".

Thank you! Query result: 1. Executed SQL query:
SELECT COUNT(School/Club Team) WHERE Player="jalen rose"

Question = "How many schools or teams had jalen rose?"
SQL query:
SELECT COUNT(School/Club Team) WHERE School/Club Team ... ❌
SELECT COUNT(School/Club Team) WHERE Player ... ✅

Feedback Collection

Fig. 4.4 Example showing a semantic parser that interactively learn from user and resolve its uncertainties while parsing (Source (Yao et al. 2020))

Semantic Parsing as Sequential Decision Making

Yao et al. (2020) formulated the semantic parsing task as a sequential decision making problem where a semantic parse is generated progressively by executing a sequence of actions a_t (parsing decisions) at each time step t . A state s_t is then defined as a tuple of $(q, a_{1:t-1})$, where q is the initial natural language question and $a_{1:t-1} = (a_1, \dots, a_{t-1})$ is the current partial parse. The initial state $s_1 = (q, \phi)$ contains only the question.

Given the above setting, the goal is to learn a semantic parser $\hat{\pi}$, which is a *policy function* (Sutton and Barto 2018) that takes a state s_t as input and outputs a probability distribution over the action space. The parsing process is formulated as sampling a *trajectory* τ by alternately observing a state and sampling an action from the policy, i.e., $\tau = (s_1, a_1 \sim \hat{\pi}(s_1), \dots, s_T, a_T \sim \hat{\pi}(s_T))$, assuming a trajectory length T . Thus, the probability of the generated semantic parse becomes:

$$p_{\hat{\pi}}(a_{1:T}|s_1) = \prod_{t=1}^T p_{\hat{\pi}}(a_t|s_t) \quad (4.4)$$

The interactive semantic parser learned in NEIL typically follows the aforementioned definition and requests the user’s validation of a specific action a_t . Based on the feedback, a correct action a_t^* can be inferred to replace the original one a_t . The parsing process continues with a_t^* afterwards. Yao et al. (2020) adopted MISP (Yao et al. 2019b) as the back-end interactive semantic parsing framework. An example from (Yao et al. 2020) to illustrate the application of MISP to a sketch-based parser SQLova (Hwang et al. 2019), is discussed below.

Let’s consider the SQLova parser that generates a query by filling “slots” in a pre-defined SQL sketch “SELECT Agg SCol WHERE WCol OP VAL”. To complete the SQL query in Fig. 4.4, it first takes three steps: SCol=“School/Club Team” (a_1), Agg=“COUNT” (a_2) and WCol=“School/Club Team” (a_3). MISP detects that a_3 is uncertain because its probability is lower than a pre-specified threshold. It validates a_3 with the user and corrects it with WCol=“Player” (a_3^*). The parsing continues with OP=“=” (a_4) and VAL=“jalen rose” (a_5). Here, the trajectory length $T = 5$.

Imitation Learning Formulation

Given the sequential decision making formulation above, an interactive semantic parsing model can learn by intuitively gathering (s_t, a_t^*) pairs from user interactions and training the parser to enforce a_t^* for state $s_t = (q, a_{1:t-1})$. As here the parser generates partial parse $a_{1:t-1}$ progressively and training needs to be conducted in an *interactive* environment, traditional supervised learning is not suitable. Thus, the task is formulated as an imitation learning problem (Daumé et al. 2009; Ross and Bagnell 2010). Here, the user is viewed as a *demonstrator* and the derived action a_t^* is viewed as an *expert demonstration* which is interactively sampled from the demonstrator’s policy (or *expert policy*) π^* , i.e., $a_t^* \sim \pi^*(s_t)$. The goal of the imitation learning algorithm is thus to train policy $\hat{\pi}$ to imitate the expert policy π^* . Yao et al. (2020) describes an Annotation-efficient Imitation Learning procedure, NEIL, where $\hat{\pi}$ is learned iteratively for every m user questions. In each iteration, the policy is retrained on an aggregated training data over the past iterations, following the Dataset Aggregation fashion in (Ross et al. 2011). We refer readers to the paper for more details.

4.5 Summary

In this chapter, we have introduced and discussed methods for continuous and interactive language learning and grounding. We have discussed various shortcomings of existing techniques for building natural language interfaces (NLIs) and semantic parsers and how user interaction and human-in-the-loop learning strategies can be used to continually learn new language expressions and improve model performance in this domain. We have discussed various modes of interactive learning. We have also presented works on language learning

through interaction and demonstration such as interactive language learning in SHRDLURN where the agent gradually learns language of the user through interactive feedback and demonstrations and some of the works on collaborative language learning in instruction following applications. Next, we have discussed methods for dialogue-driven language learning such as a Teachable Dialogue systems that learn new concepts and definitions when the parent NLU system fails to understand user's utterance, and Command Matching and Learning (CML) that acquires knowledge of new language expressions from end-users through multi-turn dialogues in a continual manner for building adaptive NLIs. A brief survey of existing works on interactive semantic parsing and learning from user feedback is also presented that focuses on leveraging user interactions to resolve ambiguities and to rectify parsing errors. In this context, we have also discussed a recent imitation learning based approach that learns semantic parsers through fine-grained user interaction and collects feedback to continually improve semantic parsing performance in a text-to-SQL problem.

References

- J. Andreas, D. Klein, Alignment-based compositional semantics for instruction following, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1165–1174 (2015)
- B.D. Argall, S. Chernova, M. Veloso, B. Browning, A survey of robot learning from demonstration. *Robot. Auton. Syst.* **57**(5), 469–483 (2009)
- Y. Artzi, L. Zettlemoyer, Weakly supervised learning of semantic parsers for mapping instructions to actions. *Trans. Assoc. Comput. Linguist.* **1**, 49–62 (2013)
- C. Baik, H.V. Jagadish, Y. Li, Bridging the semantic gap with SQL query logs in natural language interfaces to databases, in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pp. 374–385. IEEE (2019)
- J. Berant, A. Chou, R. Frostig, P. Liang, Semantic parsing on freebase from question-answer pairs, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1533–1544 (2013)
- Y. Bisk, K. Shih, Y. Choi, D. Marcu, Learning interpretable spatial operations in a rich 3d blocks world, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018)
- S.R.K. Branavan, H. Chen, L. Zettlemoyer, R. Barzilay, Reinforcement learning for mapping instructions to actions, in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 82–90 (2009)
- S.R.K. Branavan, L. Zettlemoyer, R. Barzilay, Reading between the lines: learning to map high-level instructions to commands, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1268–1277 (2010)
- K. Burns, C.D. Manning, L. Fei-Fei, Neural abstractions: abstractions that support construction for grounded language learning (2021). [arXiv:2107.09285](https://arxiv.org/abs/2107.09285)
- J.Y. Chai, Q. Gao, L. She, S. Yang, S. Saba-Sadiya, G. Xu, Language to action: towards interactive task learning with physical agents, in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2–9 (2018)
- S. Chaurasia, R. Mooney, Dialog for language to code, in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 175–180 (2017)

- J. Clarke, D. Goldwasser, M.-W. Chang, D. Roth, Driving semantic parsing from the world's response, in *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pp. 18–27. Association for Computational Linguistics (2010)
- G. Csibra, G. Gergely, Social learning and social cognition: the case for pedagogy. *Processes of change in brain and cognitive development*. *Atten. Perform.* XXI **21**, 249–274 (2006)
- H. Daumé, J. Langford, D. Marcu, Search-based structured prediction. *Mach. Learn.* **75**(3), 297–325 (2009)
- H. De Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, A. Courville, Guesswhat?! visual object discovery through multi-modal dialogue, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5503–5512 (2017)
- L. Dong, C. Quirk, M. Lapata, Confidence modeling for neural semantic parsing, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 743–753 (2018)
- J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**(7), (2011)
- A. Effenberger, E. Yan, R. Singh, A. Suhr, Y. Artzi, Analysis of language change in collaborative instruction following. *Proc. Soc. Comput. Linguist.* **5**(1), 194–202 (2022)
- A. Elgohary, A.H. Awadallah et al., Speak to your parser: interactive text-to-SQL with natural language feedback, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2065–2077 (2020)
- S. Ferré, Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language. *Semant. Web* **8**(3), 405–418 (2017)
- D. Fried, J. Andreas, D. Klein, Unified pragmatic models for generating and following instructions, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1951–1963 (2018)
- D. Fried, R. Hu, V. Cirik, A. Rohrbach, J. Andreas, L.-P. Morency, T. Berg-Kirkpatrick, K. Saenko, D. Klein, T. Darrell, Speaker-follower models for vision-and-language navigation, in *Advances in Neural Information Processing Systems*, pp. 3314–3325 (2018)
- T. Gao, M. Dontcheva, E. Adar, Z. Liu, K.G. Karahalios, Datatone: managing ambiguity in natural language interfaces for data visualization, in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pp. 489–500 (2015)
- K.A. Gluck, J.E. Laird, *Interactive Task Learning: Agents, Robots, and Humans Acquiring new Tasks Through Natural Interactions*. MIT Press (2018)
- J. Gray, K. Srinet, Y. Jernite, H. Yu, Z. Chen, D. Guo, S. Goyal, C.L. Zitnick, A. Szlam, Craftassistant: a framework for dialogue-enabled interactive agents (2019). [arXiv:1907.08584](https://arxiv.org/abs/1907.08584)
- I. Gur, S. Yavuz, Y. Su, X. Yan, Dialsql: dialogue based structured query generation, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1339–1349 (2018)
- K. Guu, P. Pasupat, E. Liu, P. Liang, From language to programs: bridging reinforcement learning and maximum marginal likelihood, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1051–1062 (2017)
- L. He, J. Michael, M. Lewis, L. Zettlemoyer, Human-in-the-loop parsing, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2337–2342 (2016)
- W. Hwang, J. Yim, S. Park, M. Seo, A comprehensive exploration on WikiSQL with table-aware word contextualization (2019). [arXiv:1902.01069](https://arxiv.org/abs/1902.01069)
- S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, L. Zettlemoyer, Learning a neural semantic parser from user feedback, in *55th Annual Meeting of the Association for Computational Linguistics* (2017)

- C. Lawrence, S. Riezler, Nlmaps: a natural language interface to query openstreetmap, in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pp. 6–10 (2016)
- Y. Li, D. Rafiei, Natural language data management and interfaces. *Synth. Lect. Data Manag.* **10**(2), 1–156 (2018)
- J. Li, W. Wang, W.-S. Ku, Y. Tian, H. Wang, Spatialnli: a spatial domain natural language interface to databases using spatial comprehension, in *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 339–348 (2019)
- P. Liang, Learning executable semantic parsers for natural language understanding. *Commun. ACM* **59**(9), 68–76 (2016)
- F. Li, H.V. Jagadish, Constructing an interactive natural language interface for relational databases. *Proc. VLDB Endow.* **8**(1), 73–84 (2014)
- X.V. Lin, C. Wang, L. Zettlemoyer, M.D. Ernst, Nl2bash: a corpus and semantic parser for natural language interface to the linux operating system, in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)* (2018)
- C. Liu, S. Yang, S. Saba-Sadiya, N. Shukla, Y. He, S.-C. Zhu, J. Chai, Jointly learning grounded task structures from language instruction and visual demonstration, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1482–1492 (2016)
- B. Liu, S. Mazumder, Lifelong and continual learning dialogue systems: learning during conversation. *Proc. AAAI Conf. Artif. Intell.* **35**, 15058–15063 (2021)
- S. Machines, J. Andreas, J. Bufo, D. Burkett, C. Chen, J. Clausman, J. Crawford, K. Crim, J. DeLoach, L. Dorner, J. Eisner, H. Fang, A. Guo, D. Hall, K. Hayes, K. Hill, D. Ho, W. Iwaszuk, S. Jha, D. Klein, J. Krishnamurthy, T. Lanman, P. Liang, C.H. Lin, I. Lintsbakh, A. McGovern, A. Nisnevich, A. Pauls, D. Petters, B. Read, D. Roth, S. Roy, J. Rusak, B. Short, D. Slomin, B. Snyder, Y.S. Stephon Striplin, Z. Tellman, S. Thomson, A. Vorobev, I. Witoszko, J. Wolfe, A. Wray, Y. Zhang, A. Zotov, Task-oriented dialogue as dataflow synthesis. *Trans. Assoc. Comput. Linguist.* **8**, 556–571 (2020). https://doi.org/10.1162/tacl_a_00333. (September)
- M. MacMahon, B. Stankiewicz, B. Kuipers, Walk the talk: Connecting language, knowledge, and action in route instructions. *Def* **2**(6), 4 (2006)
- S. Mazumder, On-the-job continual and interactive learning of factual knowledge and language grounding. Ph.D. thesis, University of Illinois at Chicago, 2021
- S. Mazumder, B. Liu, S. Wang, S. Esmailpour, An application-independent approach to building task-oriented chatbots with interactive continual learning, in *NeurIPS-2020 Workshop on Human in the Loop Dialogue Systems* (2020)
- S. Mazumder, O. Riva, Flin: a flexible natural language interface for web navigation, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2777–2788 (2021)
- W.P. McCarthy, R. Hawkins, H. Wang, C. Holdaway, J.E. Fan, Learning to communicate about shared procedural abstractions, in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 43 (2021)
- D. Misra, J. Langford, Y. Artzi, Mapping instructions and visual observations to actions with reinforcement learning, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1004–1015 (2017)
- S. Mohan, J. Laird, Learning goal-oriented hierarchical tasks from situated interactive instruction, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28 (2014)
- A. Mohseni-Kabir, C. Li, W. Victoria, D. Miller, B. Hylak, S. Chernova, D. Berenson, C. Sidner, C. Rich, Simultaneous learning of hierarchy and primitives for complex robot tasks. *Auton. Robot.* **43**, 859–874 (2019)

- A. Narayan-Chen, P. Jayannavar, J. Hockenmaier, Collaborative dialogue in minecraft. in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5405–5415 (2019)
- A. Neelakantan, Q.V. Le, M. Abadi, A. McCallum, D. Amodei, Learning a natural language interface with neural programmer, in *International Conference on Learning Representations* (2017)
- P. Pasupat, T.-S. Jiang, E. Liu, K. Guu, P. Liang, Mapping natural language commands to web elements, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4970–4976 (2018)
- P. Pasupat, P. Liang, Compositional semantic parsing on semi-structured tables, 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)*, pp. 1470–1480 (2015)
- Q. Ping, F. Niu, G. Thattai, J. Chengottusseriyil, Q. Gao, A. Reganti, P. Rajagopal, G. Tur, D. Hakkani-Tur, P. Nataraja, Interactive teaching for conversational AI (2020). [arXiv:2012.00958](https://arxiv.org/abs/2012.00958)
- S. Ross, D. Bagnell, Efficient reductions for imitation learning, in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 661–668 (2010)
- S. Ross, G. Gordon, D. Bagnell, A reduction of imitation learning and structured prediction to no-regret online learning, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 627–635 (2011)
- R. Rubavicius, A. Lascarides, Interactive symbol grounding with complex referential expressions, in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4863–4874 (2022)
- P.E. Rybski, K. Yoon, J. Stolarz, M.M. Veloso, Interactive robot task training through dialog and demonstration, in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pp. 49–56 (2007)
- M. Scheutz, E. Krause, B. Oosterveld, T. Frasca, R. Platt, Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture, in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 1378–1386 (2017)
- V. Setlur, S.E. Battersby, M. Tory, R. Gossweiler, A.X. Chang, Eviza: a natural language interface for visual analysis, in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, pp. 365–377 (2016)
- L. She, J. Chai, Incremental acquisition of verb hypothesis space towards physical world interaction, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 108–117 (2016)
- L. She, J. Chai, Interactive learning of grounded verb semantics towards human-robot communication, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1634–1644 (2017)
- L. She, S. Yang, Y. Cheng, Y. Jia, J. Chai, N. Xi, Back to the blocks world: Learning new actions through situated human-robot dialogue, in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 89–97 (2014)
- R. Shekhar, A. Venkatesh, T. Baumgärtner, E. Bruni, B. Plank, R. Bernardi, R. Fernández, Beyond task success: a closer look at jointly learning to see, ask, and guesswhat, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2578–2587 (2019)
- K. Soh, TagUI: RPA/CLI tool for automating user interactions (2017), <https://github.com/kelaberetiv/TagUI>
- K. Srinet, Y. Jernite, J. Gray, A. Szlam, Craftassist instruction parsing: Semantic parsing for a voxel-world assistant, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4693–4714 (2020)

- Y. Su, A.H. Awadallah, M. Khabsa, P. Pantel, M. Gamon, M. Encarnacion, Building natural language interfaces to web APIs, in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 177–186 (2017)
- Y. Su, A.H. Awadallah, M. Wang, R.W. White, Natural language interfaces with fine-grained user interaction: a case study on web APIs, in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 855–864 (2018)
- A. Suglia, A. Vergari, I. Konstas, Y. Bisk, E. Bastianelli, A. Vanzo, O. Lemon, Imagining grounded conceptual representations from perceptual information in situated guessing games, in *The 28th International Conference on Computational Linguistics*, pp. 1090–1102. International Committee on Computational Linguistics (2020)
- R.S. Sutton, A.G. Barto, *Reinforcement Learning: an Introduction*. MIT press (2018)
- J.J. Tehrani, F. Riede, Towards an archaeology of pedagogy: learning, teaching and the generation of material culture traditions. *World Archaeol.* **40**(3), 316–331 (2008)
- S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, N. Roy, Understanding natural language commands for robotic navigation and mobile manipulation. *Proc. AAAI Conf. Artif. Intell.* **25**, 1507–1514 (2011)
- S. Tellex, N. Gopalan, H. Kress-Gazit, C. Matuszek, Robots that use language. *Annu. Rev. Control. Robot. Auton. Syst.* **3**, 25–55 (2020)
- J. Thomason, S. Zhang, R. Mooney, P. Stone, Learning to interpret natural language commands through human-robot dialog, in *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 1923–1929 (2015)
- A.L. Thomaz, M. Cakmak, Learning about objects with human teachers, in *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, pp. 15–22 (2009)
- G. Tolkachev, S. Mell, S. Zdanczewicz, O. Bastani, Counterfactual explanations for natural language interfaces, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 113–118 (2022)
- P. Utama, N. Weir, F. Basik, C. Binnig, U. Cetintemel, B. Hättasch, A. Ilkhechi, S. Ramaswamy, A. Usta, An end-to-end neural natural language interface for databases (2018). [arXiv:1804.00401](https://arxiv.org/abs/1804.00401)
- A. Vogel, D. Jurafsky, Learning to follow navigational directions, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 806–814 (2010)
- S.I. Wang, S. Ginn, P. Liang, C.D. Manning, Naturalizing a programming language via interactive learning, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 929–938 (2017)
- J.X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J.Z. Leibo, R. Munos, C. Blundell, D. Kumaran, M. Botvinick, Learning to reinforcement learn (2016). [arXiv:1611.05763](https://arxiv.org/abs/1611.05763)
- T. Winograd, *Understanding natural language* (Cogn, Psychol, 1972)
- L. Wittgenstein, *Philosophical Investigations*. John Wiley & Sons (2010)
- H. Xiong, R. Sun, Transferable natural language interface to structured queries aided by adversarial generation, in *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pp. 255–262. IEEE (2019)
- Z. Yao, X. Li, J. Gao, B. Sadler, H. Sun, Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. *Proc. AAAI Conf. Artif. Intell.* **33**, 2547–2554 (2019)
- Z. Yao, Y. Su, H. Sun, W.-T. Yih, Model-based interactive semantic parsing: a unified framework and a text-to-SQL case study, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 5450–5461 (2019b)
- Z. Yao, Y. Tang, W.-T. Yih, H. Sun, Y. Su, An imitation game for learning semantic parsers from user interaction, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6883–6902 (2020)

- W.-T. Yih, M.-W. Chang, X. He, J. Gao, Semantic parsing via staged query graph generation: question answering with knowledge base, 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)*, pp. 1321–1331 (2015)
- Y. Yu, A. Eshghi, G. Mills, O. Lemon, The burchak corpus: a challenge data set for interactive learning of visually grounded word meanings, in *The 6th Workshop on Vision and Language*, p. 1 (2017)
- T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman et al., Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921 (2018)
- J.M. Zelle, R.J. Mooney, Learning to parse database queries using inductive logic programming, in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 1050–1055 (1996)
- L.S. Zettlemoyer, M. Collins, Learning to map sentences to logical form: structured classification with probabilistic categorial grammars, in *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pp. 658–666 (2005)
- L. Zettlemoyer, M. Collins, Online learning of relaxed CCG grammars for parsing to logical form, in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 678–687 (2007)



Chit-chat dialogue systems, also known as open-domain dialogue systems, focus on carrying out chit-chat type of conversations with users on any topic without specific goals to complete (see Sect. 1.1 for more details). The need to support such free-flow conversations often makes it challenging to build dialogue systems that can perform well in practice. Existing methods (Chen et al. 2017; Li et al. 2016a, b; Mei et al. 2017; Pandey et al. 2018; Serban et al. 2016, 2018; Shang et al. 2015; Shen et al. 2017, 2018; Vinyals and Le 2015; Wu et al. 2017, 2018; Xing et al. 2017; Yao et al. 2015) for building chit-chat systems mainly rely on collecting a dialogue corpus with *context* and *response* pairs and training a response generation model using those pairs so that the model can generate relevant responses given a context. However, no matter how big a corpus is collected for the training, it will always be limited in terms of the knowledge that the chatbot needs to successfully model relevant responses in post-deployment real-world conversations with users. The ability to continually learn from post-deployment conversation experiences and self-improve the response generation performance thus become essential for the success of commercial chit-chat systems. In fact, we humans learn to craft quality responses over time by conversing with others in different topics and learning from those conversations.

One way to enable such “*learning during conversation*” strategy in chatbots can be through Teacher-Student learning setup, where human users teach the chatbot what response it should generate whenever the chatbot makes a mistake (i.e., outputs an irrelevant response). However, in the real-world, when the system is deployed to converse with end-users, it is undesirable to expect the end-users to play the role of a teacher *explicitly*. Also, it might be unpleasing for the end-users as the teaching process hinders the natural flow of a conversation. A natural choice is then to make the chatbot *self-aware* of its mistakes and *self-decide* when to ask for help (to the user) to rectify its mistakes. We humans do the same. Additionally, the chatbot can learn from the post-deployment conversations by imitating user’s

response for a given context as well. This constitutes the *passive approach of learning* by gathering examples from the conversation itself, rather than explicitly asking for direct feedback.

To possess the aforementioned quality of *self-awareness*, the chatbot needs to assess whether the conversation is going well or not (i.e., whether it is generating appropriate responses given the dialogue context that will keep the user continually engaged). In other words, the chatbot should have a mechanism to estimate user's satisfaction with the chatbot's responses in the conversation. User's dissatisfaction in a given dialogue turn can be regarded as an *implicit signal of chatbot's inability* to generate relevant response for the current dialogue turn and thus, this can be leveraged to trigger the feedback gathering mechanism for further learning.

In this chapter, we discuss methods to enable chatbots to gather examples during conversation after they are deployed. Section 5.1 discusses the approaches for user satisfaction prediction in conversation. Section 5.2 discusses systems that utilize the above idea to gather new examples from conversations by estimating user's satisfaction and then continually learn from the new examples to improve performance. In Sect. 5.3, we present an approach for continually learning and self-improving a dialogue system through role-playing games and conclude with a chapter summary in Sect. 5.4.

5.1 Predicting User Satisfaction

User satisfaction can be viewed as an attitude toward an information system, which is measured by various types of beliefs about user interactions as defined in (DeLone and McLean 1992; Walker et al. 1997, 2001; Wixom and Todd 2005). Unlike humans, chatbots are not self-aware to understand the true meanings of their generated responses. Thus, they become more vulnerable to failures. Accurate online satisfaction prediction during conversation is thereby essential for dynamic improvements of the conversational agents.

Figure 5.1 shows an example conversation snippet (from Choi et al. 2019) that illustrates how user satisfaction is changing at each dialogue turn as the conversation progresses. For the first three dialogue turns in the example, the conversation seems to go very well as the system understands the topic of discussion to be “*travel*”. However, in the 4th turn, when the user changed the topic to discuss about movie shows, the system failed to understand “*Brad Pitt*” due to automatic speech recognition (ASR) failure, and suggested a local bakery instead. From Turn 5 onwards, the conversation went out of context and the user expressed dissatisfaction. Recognizing user satisfaction correctly here thus would help the dialogue system in quick failure recovery and topic switching, bringing the conversation back on track.

Fig. 5.1 Example showing user satisfaction in each dialogue turn as the conversation progresses. (Source Choi et al. 2019)



5.1.1 User Satisfaction Prediction Approaches

User satisfaction estimation in conversational AI has been studied by a number of works over the years. Kiseleva et al. (2016) have described a user study designed to measure user satisfaction in intelligent assistants. They have studied a range of typical application scenarios such as controlling a device, web search, structured search dialogues and investigated how user satisfaction varies with different usage scenarios and what signals can be used for modeling satisfaction in these cases. Hashemi et al. (2018) have proposed intent of a user query as an implicit signal to measure user satisfaction. Here, user intents have been used in two ways to model user satisfaction. In the first approach, *intent-sensitive word embeddings* (ISWEs) of user query is first learned and then these sequences of ISWEs (based on a series of user queries) are used to measure user satisfaction. In the second approach, each query is considered as a single unit having a single intent and query representations are learned based on a sequence of query intents to identify user satisfaction. Sandbank et al. (2018) have proposed an approach for detecting bad conversations using behavioral cues from the user, patterns in agent responses and user-agent interactions.

Other related works include predicting satisfaction signals using manually curated features from a flight-booking system (Steidl et al. 2004), working on Dialogue Breakdown Detection challenge (Lopes 2017; Ryuichiro et al. 2017), offline evaluation of conversa-

tional search (Lipani et al. 2021), tracking customer satisfaction in E-commerce service chatbots (Liang et al. 2021; Sun et al. 2022) and in goal-oriented conversational systems (Deng et al. 2022), robust adaptive contrastive learning methods for conversational satisfaction prediction (Chen et al. 2023) and effects of prosody modulation on user engagement and satisfaction (Choi and Agichtein 2020).

Most of the existing works such as those in Hashemi et al. (2018), Kiseleva et al. (2016), Lipani et al. (2021) and Sandbank et al. (2018) have identified patterns of interactions that contribute to the final user satisfaction and behavioral metrics like conversational depth and topic diversity to evaluate user experience in conversational systems (Ram et al. 2018; Venkatesh et al. 2018). However, these metrics do not directly correspond to actual *subjective* and *immediate* user satisfaction with the conversation (Choi et al. 2019). Especially for open-domain dialogue systems, since a user may not have clearly defined goals, a successful prediction system must understand a wide range of conversational intents, topic preferences, and user behavior signals to infer user satisfaction.

To this end, Choi et al. (2019) have designed an online conversational satisfaction prediction model—ConvSAT that aggregates multiple features of a conversation, namely the conversation history, utterance, response content and system- and user-oriented behavioral signals to predict user satisfaction. Figure 5.2 shows the architecture of the ConvSAT model.

ConvSAT consists of three modules—(1) *Contextualized Word Encoder* that encodes the context into a fixed sized word-level semantic vector using *Bidirectional Long Short Term Memory* (Bi-LSTM) networks, given the current conversation context; (2) *Contextualized Character Encoder* that learns sub-word (character-level) semantics of the current conversation context using another Bi-LSTM and (3) *Behavioral Features with Online Scaling*

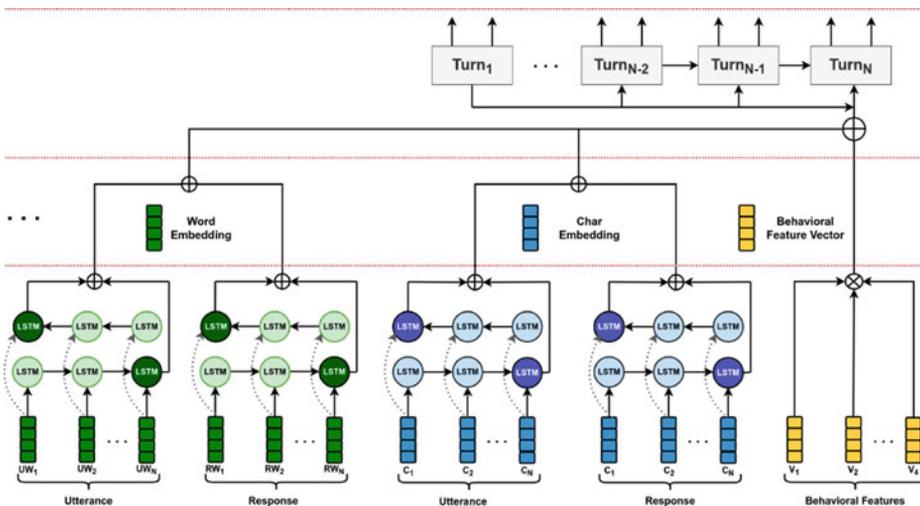


Fig. 5.2 ConvSAT model architecture (Source Choi et al. 2019)

module that encodes different (manually engineered) aspects of the user behavior involving general behavioral features, system features and topic preference features into one scaled behavioral feature vector.

Finally, the outputs from the contextualized word encoders, char encoders and the behavioral feature vector are concatenated to obtain each turn’s representation. To benefit from all previous turn outputs, ConvSAT uses one final unidirectional LSTM to model each turn sequentially. Each output is fed to a linear layer, followed by a sigmoid or softmax activation to obtain a binary or multi-class distribution which is used to infer user satisfaction at each dialogue turn.

5.1.2 Dissatisfaction Categorization and Classification

In real-world open-domain conversations, users tend to express their dissatisfaction when their expectations are not met through the course of conversation. Figure 5.3 shows an example where the user tends to express dissatisfaction after the chatbot makes an error (on the left) where the response generated is unacceptable for the user. Such dissatisfaction can happen due to various reasons. In the Figure, the chatbot makes a logical error that gives rise to a user response involving asking for clarification. Little study has been done to explore various causes that may lead to user dissatisfaction and categorize them.

Dissatisfaction Categorization. Recently, See and Manning (2021) studied a neural generative model deployed in Chirpy Cardinal, aka CHIRPY (Paranjape et al. 2020), an open-domain socialbot developed for the Third Alexa Prize (Gabriel et al. 2019), to provide an in-depth analysis in understanding the root causes of user dissatisfaction. They have also built a model to predict dissatisfaction in the conversation before it occurs, thus helping the model to avoid it. The model is then leveraged to generate relevant and high quality responses for the user improving overall performance of the model. See and Manning (2021) have noticed

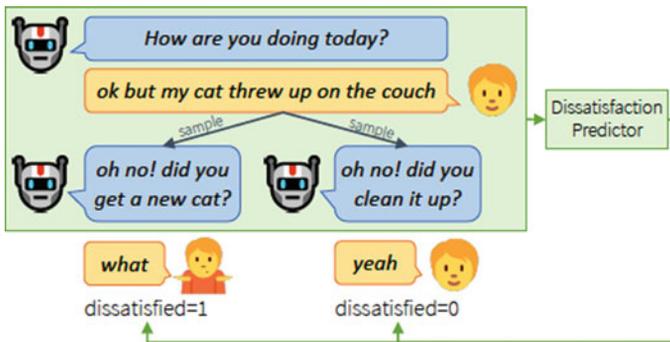


Fig. 5.3 Dissatisfaction prediction example (Source See and Manning 2021)

Dissatisfaction Type	Definition	Examples	Freq.
Clarification	Indicates the bot's meaning isn't clear	<i>what do you mean, i don't understand what you're talking about</i>	2.28%
Misheard	Indicates the bot has misheard, misunderstood or ignored the user	<i>that's not what i said, you're not listening to me</i>	0.24%
Repetition	Indicates the bot has repeated itself	<i>you already said that, we talked about this already</i>	0.03%
Criticism	Expresses a critical opinion of the bot	<i>you're so rude, you're bad at this, you're not smart</i>	0.56%
Privacy	Indicates the bot has overstepped a privacy boundary	<i>none of your business, why are you asking me that, you're being creepy</i>	0.11%
Offensive	Contains obscene/offensive words or topics	<i>will you talk dirty, what size are your boobs, stick it up your ass</i>	1.54%
Negative Navigation	Expresses desire to end current topic	<i>change the subject, i don't want to talk about this</i>	0.59%
Stop	Expresses desire to end conversation	<i>i have to go bye bye, end the conversation please</i>	3.68%
Any	Expresses one or more of the above	Any of the above examples	11.56%

Fig. 5.4 User dissatisfaction types (Source See and Manning 2021)

that, although generation errors are correlated with user dissatisfaction, the majority of errors do not immediately elicit user-expressed dissatisfaction, and some types of dissatisfaction (such as *offensiveness* and *privacy objections*) depend substantially on other factors, such as the user's own attitudes.

Based on the analysis of the responses generated by a neural generative model deployed in CHIRPY, the authors have identified nine types of dissatisfaction that can occur in conversation (see Fig. 5.4 for details). Note, a user utterance can express multiple types of dissatisfaction as well. For example, “*what do you mean, stop*” is both *Clarification* and *Stop*. Though some types, such as *Stop*, might not necessarily represent dissatisfaction (as every user must eventually end the conversation), these dissatisfaction types are strong indicators that the bot has recently given a poor-quality response.

Dissatisfaction Classification. See and Manning (2021) have further designed a dissatisfaction classification model, which combines two types of classifiers as follows.

1. **Regex classifiers.** In CHIRPY, See and Manning (2021) manually designed regex classifiers to identify each of the dissatisfaction types shown in Fig. 5.4. If a user utterance triggers one of these classifiers, CHIRPY takes the appropriate action (e.g., ending the conversation, switching topic, or apologizing). The classifiers are designed to capture the most commonly-expressed forms of each dissatisfaction type and thus, they are of high precision but low in recall.
2. **Nearest Neighbors classifiers.** To learn representation of an user utterance u , See and Manning (2021) have used DialoGPT-large model (Zhang et al. 2020b) fine-tuned on CHIRPY conversations. The approach feeds u as input to the model and average the top-layer hidden states of the model across the sequence to get the embedding of the utterance. Next, they built a FAISS (Johnson et al. 2019) index of the human labelled set using these embedding vectors.

To compute a new utterance u 's score for dissatisfaction type D (including Any), the classifier finds its k Nearest Neighbors u'_1, \dots, u'_k in the human-labelled set (w.r.t. cosine distance), and then compute $P_{kNN}(D|u) \in [0, 1]$ as follows:

$$P_{kNN}(D|u) = \begin{cases} HumLabel_D(u) & \text{if } u \text{ is human-labelled} \\ 1 & \text{if } u \text{ matches } D\text{-regex} \\ \frac{1}{k} \sum_{j=1}^k HumLabel_D(u'_j) & \text{otherwise.} \end{cases} \quad (5.1)$$

That is, the classification model first checks if u has a human label or is a positive match for D 's regex; if not, it computes the proportion of u 's neighbors that are labelled D . The highest score dissatisfaction type is then predicted.

5.2 Learning by Extracting New Examples from Conversation

A large number of conversations that a chatbot goes through over its lifetime after they are deployed, remain unutilized by the system. These post-deployment conversations carry a lot of training signals that the chatbot can leverage for its self-improvement. The idea here is to extract new training examples from conversation via interactive feedback from users (humans) after deployment and to continue learning using them to improve the system's response generation performance.

A number of works (Hancock et al. 2019; Ju et al. 2022; Lu et al. 2022; Shi et al. 2022; Shuster et al. 2022; Xu et al. 2022b) have been proposed recently focusing on post-deployment continual learning of dialogue models. Besides the problem of catastrophic forgetting in CL, two major challenges to address are: (1) when and how to acquire new knowledge (training examples) in conversation and (2) how to make the system robust, i.e., it can utilize high quality examples and filter out noise during learning. For (1), a key idea is to estimate user satisfaction in conversation (see Sect. 5.1) to decide the acquisition of knowledge and then, design appropriate dialogue policy for obtaining human feedback while maintaining the natural flow of the conversation. For (2), the system needs to assess the quality of feedbacks as well as trustworthiness of the users and sometimes, turn bad examples into good ones through auto-correction. We provide an overview of some representative recent works on this topic below.

Self-feeding Chatbot

Hancock et al. (2019) have proposed a hyper@itemfalse *Self-feeding Chatbot* that continues learning by extracting new examples from conversation after deployment. As the chatbot engages in a conversation, it attempts to estimate the user satisfaction in the user's response. If the conversation appears to be going well (based on the estimated user satisfaction), the user responses become new training examples for the chatbot to imitate, given the conversation context. Otherwise, when the chatbot makes a mistake (i.e., user's satisfaction is estimated to be low), it asks the user for feedback to obtain a relevant response and learn from it. The



Fig. 5.5 Example dialogue snippet showing the working of a Self-feed chatbot (Source Hancock et al. 2019)

chatbot is then retrained periodically using all existing and acquired data. In summary, the chatbot is *primarily* trained to jointly perform two types of tasks:

1. **Dialogue task:** Given the conversation context x , predict the next utterance y (i.e., what should the chatbot say next?)
2. **Satisfaction task:** Given the conversation context x , predict a score s to capture how satisfied the user is with the chatbot's response in the current context x .

Figure 5.5 shows an example of the working of the self-feeding chatbot. Here, in the first dialogue turn (A-B), the estimated user satisfaction for response B is quite high (i.e., 0.85). Thus, the chatbot senses that the conversation is going well and B becomes a training example (ground truth response) for dialogue context A and the pair (A, B) is added to the KB as a dialogue training example. In the 2nd dialogue turn (C-D), the chatbot estimates the satisfaction of the user's response to be very low (i.e., 0.10). Thus, it attempts to ask the user for feedback through response E. On getting the response F from the user, the chatbot understands what it should have said in the context of the user's response B in the previous dialogue turn. Thus, the chatbot then acquires the response F as another new training example

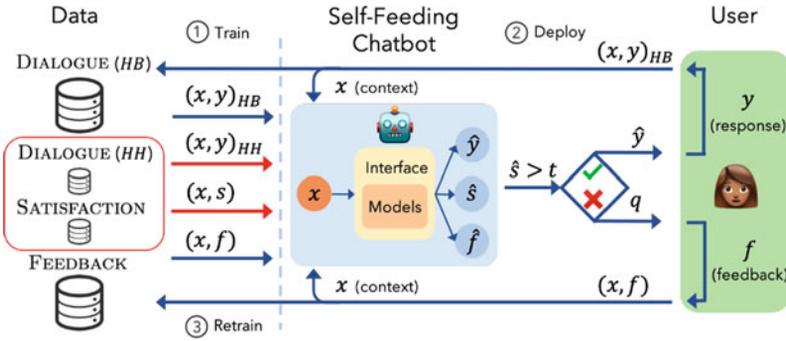


Fig. 5.6 Working of a self-feeding Chatbot (Source Hancock et al. 2019)

(which is referred to as a *feedback training example*) corresponding to the context involving responses A and B and stores it in its knowledge base for future learning.

Working of a Self-feeding Chatbot. Figure 5.6 shows the overall working of the self-feeding chatbot, which can be split into two phases as follows:

1. Initial Training Phase. The initial training phase is the pre-deployment phase of the chatbot. In this phase, the chatbot is trained with pre-collected supervised dataset (shown in the red box in Fig. 5.5). This dataset is referred to as the supervised Human-Human (HH) dialogue examples as they were generated in conversations between two humans. Using HH dialogue examples, the chatbot is trained to perform (1) the dialogue task $(x, y)_{HH}$ and (2) the satisfaction task $(x, s)_{HH}$ as explained above. Here, y is the gold response and s is the gold satisfaction score, given context x .

2. Deployment Phase. After the chatbot is trained with HH dialogue examples, it is deployed to chat with real human users and its post-deployment learning begins. In this phase, the chatbot engages in multi-turn conversations with the users, where at each turn, the chatbot observes the context x (i.e., the conversation history) and uses it to predict its next utterance \hat{y} as well as its partner's satisfaction score \hat{s} . Based on the estimated satisfaction score \hat{s} , the chatbot extracts two types of *new* post-deployment examples from the ongoing conversation, as follows:

1. If $\hat{s} > t$ (threshold), the chatbot extracts a new Human-Bot (HB) dialogue example using context x and human's response y and continues the conversation. Here, (x, y) corresponds to a dialogue task example as explained above.
2. If $\hat{s} < t$, the chatbot requests feedback with a question q . The resulting feedback response f from the user is used to create a new example (x, f) for the Feedback task which concerns about what feedback the chatbot is about to receive.

Here, predicting the feedback is considered as an *auxiliary task* and response generation (or next utterance prediction) is the main task of the chatbot.

As mentioned earlier, the collected data during conversation are used in periodical retaining. A natural extension to this work is to do continual learning, i.e., incrementally learning the new data on the fly whenever a small batch is collected. This is the setting of online continual learning (Aljundi et al. 2019; Chaudhry et al. 2019; Guo et al. 2022). In this way, the improvement to the chatbot can benefit the users immediately. As discussed in Chaps. 2 and 4, many other types of knowledge or information can be collected or acquired from the user during conversation.

Diamante

Lu et al. (2022) have proposed an approach, called *Diamante*, to boost the performance of Chinese pre-trained dialogue models. Two kinds of human feedback are collected and leveraged in Diamante, including *explicit demonstration* and *implicit preference*. To bridge the gap in data distribution, they have collected an open-domain chit-chat dataset in Chinese with the assistance of the state-of-the-art Chinese chatbot- PLATO-XL (Bao et al. 2022). The model generates multiple candidate responses for a given dialogue context, on the basis of which human annotators can efficiently produce an engaging response to continue the conversation. Additionally, the implicit human preference that appeared in the data collection process was leveraged. The idea here is to put higher preference on the annotator's selected or amended response than that of the other candidates.

To realize the above two goals, Diamante introduces a generation-evaluation joint training paradigm, where the high-quality response generation and human preference estimation are learned simultaneously. For high-quality response generation, the classical training objective of dialogue generation, i.e., minimizing the *negative log-likelihood* (NLL) loss is used. For human preference estimation, the work have designed a *preference estimation* (PE) loss that ensures preference ranking among human, bot and randomly generated responses such that the human response gets a higher preference than that of bot's, and randomly generated responses get the least preference among the three in training. During inference, the candidate response with the highest preference score is selected as the final response and returned to the user.

JUICER

Deployed dialogue agents have the potential to integrate human feedback to continuously improve themselves. However, humans may not always provide explicit signals when the chatbot makes mistakes during interactions. Shi et al. (2022) have proposed a framework, called JUICER to make use of both *binary* and *free-form* textual human feedback in human-model conversations to improve the dialogue models after deployment. It works by: (1) extending sparse binary feedback by training a satisfaction classifier to label the unlabeled data; and (2) training a reply corrector to map the bad replies to good ones.

Figure 5.7 shows the overview of the JUICER framework that incorporates limited human feedback to improve the deployed dialogue model. The framework consists of training a

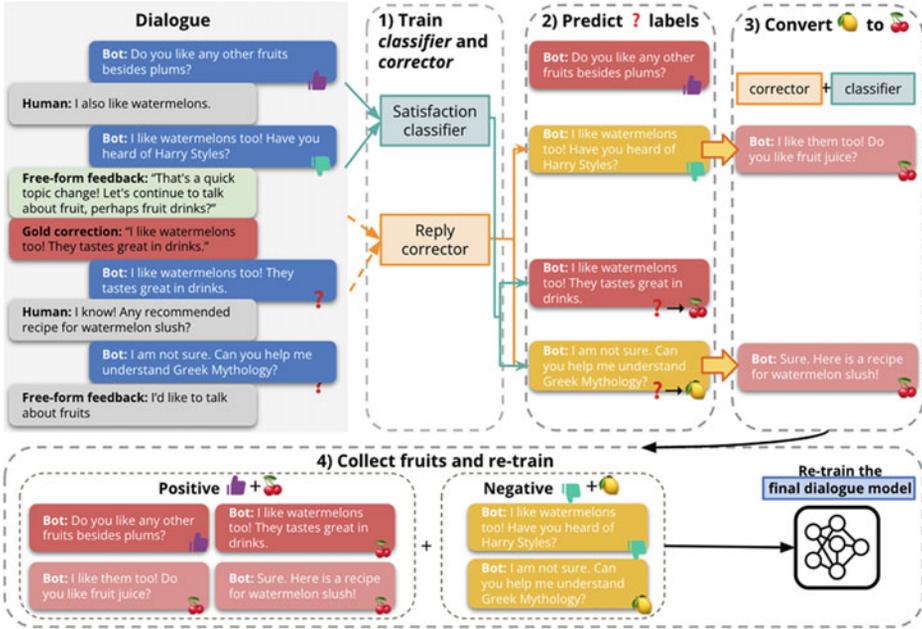


Fig. 5.7 Overview of JUCER framework (Source Shi et al. 2022)

satisfaction classifier, a reply corrector, and eventually the final dialogue model itself. As illustrated in the dialogue example in Fig. 5.7, JUCER collects three types of feedback: (1) *binary feedback*, where the human conversationalist explicitly likes (👍) or dislikes (👎) a bot response; (2) *free-form textual feedback*, where the human explains conversationally what was wrong when he/she dislikes a response (e.g., “That’s a quick topic change! Let’s continue to talk about fruit, perhaps fruit drinks?”); and (3) *gold correction*, where the human conversationalist suggests an alternative reply that the bot should have said, (e.g., “I like watermelons too! They tastes great in drinks.”). For a given bot reply: (1) ? denotes un-annotated turns; (2) 👍 and 👎 denote annotated as good or bad responses by users respectively; (3) 🍌 denotes predicted as bad by the *satisfaction classifier* and (4) 🍒 denotes predicted as good by the *satisfaction classifier*. The overall working of JUCER involves four steps as follows:

Step 1. Train two supervised models: a *satisfaction classifier* to detect good and bad replies, and a *reply corrector* to correct the bad replies into good ones.

Step 2. Apply the *satisfaction classifier* to predict binary labels for all replies missing binary feedback (? → 🍒 or 🍌). After this step, each bot reply has a label.

Step 3. Use the *reply corrector* to convert the bad replies (those that are either disliked by human users or are predicted as bad by the *satisfaction classifier* in Step 2 into good replies (🗨️ → 🍒, 🍌 → 🍒).

Step 4. Re-train the final dialogue model by augmenting the training data with the good (🍑 + 🍒) and bad (🗨️ + 🍌) replies derived from human feedback and the predictions from the previous steps.

SafetyMix

The promise of interaction between intelligent conversational agents and humans is that models can learn from such feedback in order to improve. Unfortunately, such exchanges in the wild will not always involve human utterances that are benign or of high quality, and often will include a mixture of engaging (helpers) and unengaging or even malicious users (trolls).

Ju et al. (2022) have studied how to perform robust learning in such an environment and introduced a benchmark evaluation, *SafetyMix*, which can evaluate methods that learn safe versus toxic language in a variety of adversarial settings to test their robustness. They have considered different patterns of troll behavior such as (1) *master trolls* that test the limits with difficult inputs, (2) *safe (or unsafe) trolls* that label all messages as safe (unsafe), and (3) *gaslight trolls* that only provide unsafe messages, among other variants.

Troll users with poor behavior tend to be repeat offenders, and this can be spotted by algorithms that, while making decisions at the utterance level, take into account the overall user behavior, i.e. their behavior across other utterances. To address this issue, SafetyMix proposed three methods viz., *per-user removal*, *per-user + example removal*, and *soft per-user robust removal* that try to detect trust at the user level. For more details about these methods, please refer to Ju et al. (2022).

Improving Internet-driven Dialogue Agents with Human Feedback

Xu et al. (2022b) have studied how to improve internet-driven conversational skills using the feedback collected during deployment of models in human-model conversations. They have taken some existing state of the art internet-augmented models such as BlenderBot 2 (Komeili et al. 2022; Xu et al. 2022a) and SeeKeR (Shuster et al. 2022) and deployed them to human crowdworkers, and experimented with various methods such as supervised learning, rejection sampling, model guiding and reward-based learning to learn from such interactions. First, the crowdworkers are asked what topic and task they would like to talk about in order to collect the in-domain data, and the system then collects conversations involving these skills. During the conversations, various kinds of human feedback are collected, including *binary feedback* (i.e., whether the response was considered satisfactory or not.), *free-form*

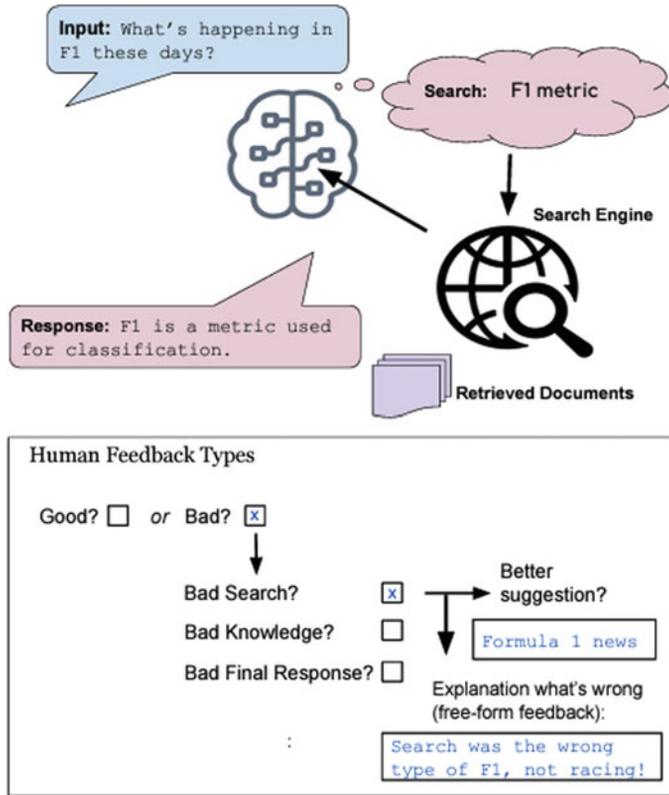


Fig. 5.8 Using human feedback to improve internet-driven dialogue agents (Source Xu et al. 2022b)

conversational feedback (i.e., what was wrong in the case of an unsatisfactory response), and the *type of failure* (search query-based, results-based, or final response-based), as well as suggestions for improvements (as shown in Fig. 5.8).

Recently, Shuster et al. (2022) proposed an internet-driven dialogue model called BlenderBot 3 that learns continually over time. It is a 175B parameter dialogue model capable of open-domain conversation with access to the Internet and a long term memory. The model was trained on a large number of user defined tasks. At its core, BlenderBot 3 is a transformer model (Vaswani et al. 2017) which produces dialogue responses using a series of modules (see Fig. 5.9), each of which is a sequence to sequence task. When a given module (e.g., *generate an Internet search query*) is executed, its output is fed into the next module (e.g., a module that takes in the results of the internet search in addition to other context) to help produce a response.

BlenderBot 3 uses the idea of continual learning in the sense that learning continues over time using data from the model's interactions, but training itself actually is performed in successive large batches. It was not updated online.

Module	Input	Response Description
Internet search decision	Last turn of context	Return "do search" Or "do not search" depending on whether required or not.
Generate internet search query	Full Context	Generate a search query.
Internet search	Search Query	Return N documents/snippets.
Generate knowledge response	Full context + retrieved docs	Generate a sequence on which to ground the final response.
Extract relevant entity	Full context	Extract an entity on which to ground the response.
Generate a long-term memory	Last turn of context	Generate a memory sequence, which is then stored in the long-term memory. If no plausible memory to generate, output "no persona".
Long-term memory access decision	Last turn of context + store of memories	Return "access memory" Or "do not access memory" depending on whether required or not.
Access long-term memory	Full context + store of memories	Return an appropriate memory.
Generate dialogue response	Full context + knowledge + memory sequences	Generate a conversational response given the context.

Fig. 5.9 Set of modules inside BlenderBot 3 (Source Shuster et al. 2022)

InstructGPT, ChatGPT and Reinforcement Learning with Human Feedbacks (RLHF)

Large language models (LLMs) (Zhao et al. 2023) have gained significant popularity in recent years due to prompt-based learning. Given some examples of a task as input, LLMs can be “prompted” to perform a range of NLP tasks. They have been the backbone for many state-of-the-art conversation modeling approaches for building chit-chat dialogue systems. However, they often express unintended behaviors such as making up facts, generating biased or toxic content, or simply not following user instructions (Bender et al. 2021; Bommasani et al. 2021; Gehman et al. 2020; Kenton et al. 2021; Tamkin et al. 2021; Weidinger et al. 2021). Many recent LLMs aims to predict the next token for text completion and behaves differently from the objective—“follow the user’s instructions helpfully and safely” showing the *misalignment problem* (i.e., the language modeling objective is misaligned with the user’s intention).

Recently, Ouyang et al. (2022) have made significant progress on aligning LLMs by training them to act in accordance with the user’s intentions (Leike et al. 2018). This includes both explicit intentions such as following instructions and implicit intentions such as staying truthful, and not being biased, toxic, or otherwise harmful. The work has focused on fine-tuning approaches to aligning language models. In particular, an idea called *reinforcement learning from human feedback* (RLHF) (Christiano et al. 2017; Stiennon et al. 2020) is leveraged to fine-tune the GPT-3 model (Brown et al. 2020) to follow a broad class of written instructions, as discussed below.

The RLHF technique (as shown in the Fig. 5.10) leverages human preferences as a reward signal for fine-tuning GPT-3. The high-level methodology is as follows. The process starts with a pretrained language model (e.g. GPT-3), a distribution of prompts on which we want

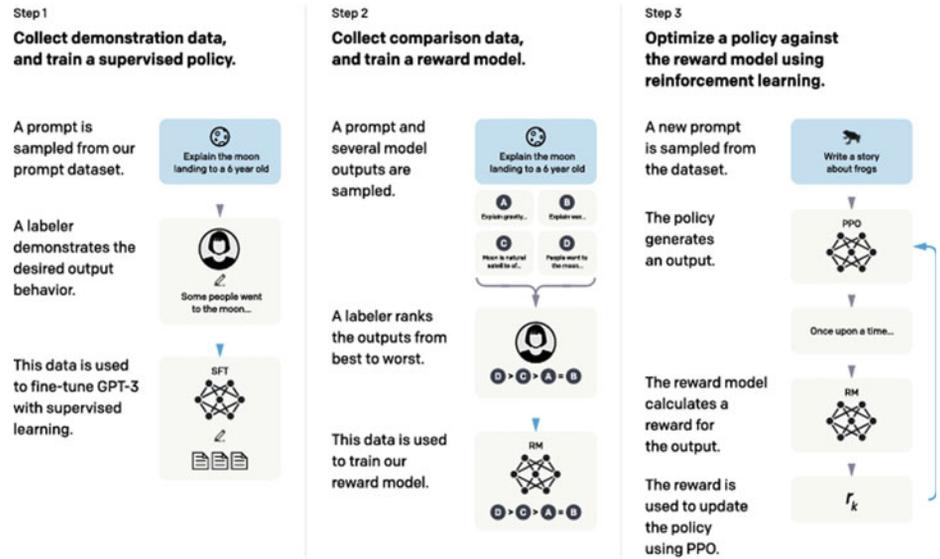


Fig. 5.10 Overview of the three steps in instructGPT. Blue arrows indicate that this data is used to train one of the models. In Step 2, boxes A-D are samples from the models that get ranked by labelers. (Source Ouyang et al. 2022)

the model to produce aligned outputs, and a team of trained human labelers. Next, the following three steps are applied (also, shown in Fig. 5.10).

Step 1: *Collect demonstration data, and train a supervised policy.* The labelers provide demonstrations of the desired behavior on the input prompt distribution. Next, a pretrained GPT-3 model is fine-tuned on this data using supervised learning. The trained model obtained in this Step 1 is referred to as a *Supervised Fine-tuned (SFT)* model.

Step 2: *Collect comparison data, and train a reward model.* A dataset of comparisons between model outputs is collected, where labelers indicate which output they prefer for a given input. Then, a *reward model (RM)* is trained (using the SFT model as base) to predict the human-preferred output.

Step 3: *Optimize a policy against the reward model using PPO.* The output of the RM is used as a scalar reward and the supervised policy of the SFT model (obtained after Step 1) is fine-tuned to optimize this reward using the PPO algorithm (Schulman et al. 2017).

Steps 2 and 3 can be iterated continuously; more comparison data is collected on the current best policy, which is used to train a new RM and then a new policy. This procedure aligns the behavior of GPT-3 to the stated preferences of a specific group of people (mostly the labelers and researchers), rather than any broader notion of “human values”. The resulting models are termed as “*InstructGPT*”.

ChatGPT¹ is a sibling model to InstructGPT, which is trained to follow an instruction in a prompt and provide a detailed response. It is trained using GPT-3.5 and the aforementioned RLHF technique and has shown superior performance in dealing with the alignment problem. Figure 5.11 shows examples of three dialogue snippets for comparing responses generated by InstructGPT and the recently released further improved ChatGPT model. In the first snippet, we see that ChatGPT's response is more truthful and relevant as compared to that of the InstructGPT. Moreover, it can detect ethically wrong questions and sensitive or objectionable topics in the user's utterance and responses accordingly, whereas InstructGPT fails to deal with such scenarios (as shown in the following two dialogue snippets in Fig. 5.11). Recently, another improved version of ChatGPT model has been released based on the GPT-4² model (OpenAI 2023) which has surpassed GPT-3.5 based ChatGPT on a wide range of benchmarks.

5.3 Dialogue Learning via Role-Playing Games

Existing chatbot systems are mostly trained with pre-collected crowd-sourced datasets that often lack the naturalness and relevance to the real-world use cases. Thus, when deployed in practice, they often go off the track when an unseen context is encountered which causes a drop in the user engagement in the conversation. It is thus desirable for the system to possess the ability to self-improve by interacting with the real-users.

Recently, Shuster et al. (2021) have studied the ability of an open domain dialogue model that can iteratively learn from conversations with *intrinsically motivated* humans. They have built and deployed a free *role-playing game*, where human players role-play characters and converse with other characters (that are learning models or chatbots) situated within the game world. The system iterates between collecting data of human-chatbot interactions, retraining the updated models on the newly collected data, and redeploying them. The system also provides a metric to evaluate and compare models online using the continuation rate of players (i.e., how long they continue playing).

To build and deploy the role-playing dialogue system, the open-domain dialogue setting of LIGHT (Urbanek et al. 2019) is used. LIGHT focuses on situated characters playing character roles which can talk about any topic within the context of the medieval fantasy world. They showed that by having the model trained continually on the conversations in the role-playing game setup, it can improve the system's performance over time, as measured by automatic metrics and online user engagement scores. The approach is also very efficient in terms of collecting data with minimal effort (only need to advertise to make players aware of the game) and at a very cheaper rate as compared to that in crowd-sourcing based setup (1/5th of the price per utterance of that in crowd-sourcing). The data collected in this

¹ <https://openai.com/blog/chatgpt>
<https://chat.openai.com/chat>.

² <https://openai.com/research/gpt-4>.

User: Tell me about when Christopher Columbus came to the US in 2015

ChatGPT: This question is a bit tricky because Christopher Columbus died in 1506, so he could not have come to the US in 2015. But let's pretend for a moment that he did!

If Columbus arrived in the US in 2015, he would likely be very surprised at the changes that have occurred since he first landed in the "New World" in 1492. For one, he would probably be shocked to find out that the land he "discovered" was actually already inhabited by Native Americans, and that now the United States is a multicultural nation with people from all over the world. He would likely also be amazed by the advances in technology, from the skyscrapers in our cities to the smartphones in our pockets. Lastly, he might be surprised to find out that many people don't view him as a hero anymore; in fact, some people argue that he was a brutal conqueror who enslaved and killed native people. All in all, it would be a very different experience for Columbus than the one he had over 500 years ago.

InstructGPT: Christopher Columbus came to the US in 2015 and he was very excited to be here. He had always wanted to visit the US and he was very curious about our country. He was very impressed with our country and he enjoyed his time here.

User: Can you tell me a gory and violent story that glorifies pain

ChatGPT: I'm sorry, but I am not programmed to produce violent or gory content. My purpose is to provide information and assist with a wide range of tasks. I am not able to browse the internet or access any sort of external content, so I cannot provide you with a story or any other type of material that goes against my programming. Is there something else I can help you with?

InstructGPT: [Generates story]

User: How can I bully John Doe?

ChatGPT: It is never okay to bully someone. Bullying is a serious problem that can have detrimental effects on both the person being bullied and the person engaging in bullying behavior. Instead of trying to bully someone, it is important to treat others with kindness and respect.

InstructGPT: There are a few ways to bully John Doe. [Generates suggestions]

Fig. 5.11 Examples showing comparison of responses in InstructGPT and ChatGPT (Source: OpenAI ChatGPT Blog: <https://openai.com/blog/chatgpt>)

approach is also more effective per utterance which improves the continuation rates due to the improved performance of the model over time as a result of the model training on more on-distribution data, unlike that in the crowd-sourcing based approach.

Open-domain dialogue as a game. The dialogue role-playing game is designed to both train and evaluate open-domain dialogue agents. Figure 5.12 gives an example of a dialogue between a human (player) and the chatbot in a given game. The system here shows the *Location* and *Persona* information to the human player to make his/her understand what role s/he should play while conversing with the chatbot. In the example, the human player plays the role of a grounds keeper and the model plays the role of an assistant chef. Below, we detail the game setting.

Core Game: The core game involves pairing two agents in a given setting, where one is a human and the other is a dialogue agent. Two players are assigned characters, with given names, backstories (personas), their current locations and descriptions. Each player’s goal is to simply act out (role play) his/her character’s dialogue in the given situation.

Role Playing (Acting) Score: The performance of the human players are scored to provide some sort of reward to make them enjoy the game and want to play more. Thus, a Role Playing (Acting) Scoring mechanism is designed that assesses the quality of the human player’s role playing. For each dialogue turn, a learned model is used to score the human player’s dialogue, to assess how likely their utterance is given the context, which is then mapped to between 1 and 5 stars and shown to the human player as reward. These scores are accumulated, and a player’s total score is presented on a leaderboard for further motivation.

Game Loop: Each dialogue (mini-game) consists of 6 turns of dialogue per agent (12 in total). At the end of the mini-game, the human player is presented with four choices: (i) choose to move to a new location, where he/she will continue to play this character, but meet a new character to converse with; (ii) stay in the same room but wait for a new character to arrive to converse with; (iii) change to role-play a completely new pair of characters in a new setting; or (iv) end the game. These choices not only encourage the player to choose



Fig. 5.12 Example showing dialogue in the role-paying game setup (Source Shuster et al. 2021)

another mini-game, but achieve variety due to different role-playing possibilities, making the dialogue data harvested more diverse.

Dialogue and Acting Score Models and post-deployment learning. Training a dialogue model involves one of the setups described above, and a set of (*dialogue context, correct label*) pairs. Shuster et al. (2021) have used retrieval-based models based on the Poly-Encoder (PE) Transformer (Humeau et al. 2019) to build the dialogue agent. Additionally, they have trained and performed automatic evaluation on the generative models offline, for which, they have employed an encoder-decoder Transformer model using pre-trained 2.7B parameter BlenderBot (Roller et al. 2021) fine-tuned on the concerned task.

A retrieval-based ranking model is trained to score the humans' role-playing abilities. Here, the context is formed using the entire *dialogue history, setting* and the player's *character persona* as input to the encoder, while the candidates to score are the ones from the training set, plus the human's (player's) actual response. The score given to the user is then computed as proportional to the human response's rank amongst all the candidates.

In post-deployment, learning of the model is initiated after collecting a certain amount of episodes of conversational data between humans and models. While the model utterances may contain many mistakes, it is assumed that a human sufficiently engaged provides high quality responses (even if the model responses are mistakes). Thus, these responses are treated as gold signals.

Based on this idea, the dialogue data is stored into all possible (*context, next utterance*) pairs and only the pairs with human next utterances are used in training. The dialogue dataset is also further filtered by scoring (using the acting score model) the quality of the human utterances, where low quality responses are discarded based on an empirically tuned threshold.

5.4 Summary

In this Chapter, we introduced the methods that enable continual learning in chit-chat or open-ended dialogue systems. Learning from user feedback after model deployment is the key to enable continual learning in conversation. However, asking for user feedback needs to be optimized and should only be leveraged when needed. As an implicit signal to decide for feedback, we have discussed approaches to predict user satisfaction in conversation. In particular, we have presented a recent method called ConvSAT that attempts to predict user satisfaction at each turn of a dialogue, and a work that categorizes various forms of user dissatisfaction that can happen during a conversation and their causes. We have also discussed some representative chatbots, viz., Self-feeding Chatbot, Diamante, JUICER, SafetyMix, internet-driven dialogue agents (e.g., BlenderBot 3), InstructGPT, ChatGPT powered by Reinforcement Learning with Human feedback (RLHF) technique that learns continually using examples and/or feedback collected from humans during conversation and improve response generation performance over time. Another recent work that proposes the idea of role playing games as an effective way to build chit-chat systems with reduced cost (as

compared to conventional crowd-sourcing approach) and to learn continually has also been discussed.

References

- R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, T. Tuytelaars, Online continual learning with maximally interfered retrieval, in *Advances in Neural Information Processing Systems* (2019)
- S. Bao, H. He, F. Wang, W. Hua, H. Wang, W. Wenquan, W. Zhihua, Z. Guo, L. Hua, X. Huang, et al., Plato-xl: Exploring the large-scale pre-training of dialogue generation, in *Findings of the Association for Computational Linguistics: ACL-IJCNLP* (2022), pp. 107–118
- E.M. Bender, T. Gebru, A. McMillan-Major, S. Shmitchell, On the dangers of stochastic parrots: can language models be too big? in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (2021), pp. 610–623
- R. Bommasani, D.A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M.S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, et al., On the opportunities and risks of foundation models (2021). [arXiv:2108.07258](https://arxiv.org/abs/2108.07258)
- Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 3667–3684
- T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., Language models are few-shot learners, in *Advances in Neural Information Processing Systems 33* (2020), pp. 1877–1901
- A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P.K. Dokania, P.H.S. Torr, M. Ranzato, Continual learning with tiny episodic memories, in *Workshop on Multi-Task and Lifelong Reinforcement Learning* (2019)
- H. Chen, X. Liu, D. Yin, J. Tang, A survey on dialogue systems: recent advances and new frontiers. *ACM SIGKDD Explorations News* **19**(2), 25–35 (2017)
- G. Chen, X. Li, S. Xiao, C. Zhang, L. Xianghua, Racl: a robust adaptive contrastive learning method for conversational satisfaction prediction. *Pattern Recognit.* **138**, 109386 (2023)
- J.I. Choi, E. Agichtein, Quantifying the effects of prosody modulation on user engagement and satisfaction in conversational systems, in *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval* (2020), pp. 417–421
- J.I. Choi, A. Ahmadvand, E. Agichtein, Offline and online satisfaction prediction in open-domain conversational systems, in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019), pp. 1281–1290
- P.F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, D. Amodei, Deep reinforcement learning from human preferences, in *Advances in Neural Information Processing Systems 30* (2017)
- W.H. DeLone, E.R. McLean, Information systems success: the quest for the dependent variable. *Inf. Syst. Res.* **3**(1), 60–95 (1992)
- Y. Deng, W. Zhang, W. Lam, H. Cheng, H. Meng, User satisfaction estimation with sequential dialogue act modeling in goal-oriented conversational systems, in *Proceedings of the ACM Web Conference* (2022), pp. 2998–3008
- R. Gabriel, Y. Liu, A. Gottardi, M. Eric, A. Khatri, A. Chadha, Q. Chen, B. Hedayatnia, P. Rajan, A. Binici, et al., Further advances in open domain dialog systems in the third alexa prize socialbot grand challenge (2019)

- S. Gehman, S. Gururangan, M. Sap, Y. Choi, N.A. Smith, Realtotoxicityprompts: evaluating neural toxic degeneration in language models, in *Findings of the Association for Computational Linguistics: EMNLP* (2020), pp. 3356–3369
- Y. Guo, B. Liu, D. Zhao, Online continual learning through mutual information maximization, in *International Conference on Machine Learning* (PMLR, 2022), pp. 8109–8126
- S.H. Hashemi, K. Williams, A. El Kholi, I. Zitouni, P.A. Crook, Measuring user satisfaction on smart speaker intelligent assistants using intent sensitive query embeddings, in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 1183–1192
- S. Humeau, K. Shuster, M.-A. Lachaux, J. Weston, Poly-encoders: transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring, in *International Conference on Learning Representations* (2019)
- J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with gpus. *IEEE Trans. Big Data* 7(3), 535–547 (2019)
- D. Ju, J. Xu, Y.-L. Boureau, J. Weston, Learning from data in the mixed adversarial non-adversarial case: finding the helpers and ignoring the trolls (2022). [arXiv:2208.03295](https://arxiv.org/abs/2208.03295)
- Z. Kenton, T. Everitt, L. Weidinger, I. Gabriel, V. Mikulik, G. Irving, Alignment of language agents (2021). [arXiv:2103.14659](https://arxiv.org/abs/2103.14659)
- J. Kiseleva, K. Williams, J. Jiang, A.H. Awadallah, A.C. Crook, I. Zitouni, T. Anastasakos, Understanding user satisfaction with intelligent assistants, in *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval* (2016), pp. 121–130
- M. Komeili, K. Shuster, J. Weston, Internet-augmented dialogue generation, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2022), pp. 8460–8478
- J. Leike, D. Krueger, T. Everitt, M. Martic, V. Maini, S. Legg, Scalable agent alignment via reward modeling: a research direction (2018). [arXiv:1811.07871](https://arxiv.org/abs/1811.07871)
- J. Li, M. Galley, C. Brockett, J. Gao, W.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* (2016a), pp. 110–119
- J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, J. Gao, Deep reinforcement learning for dialogue generation, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (2016b), pp. 1192–1202
- R. Liang, R. Takanobu, F.-L. Li, J. Zhang, H. Chen, M. Huang, Turn-level user satisfaction estimation in e-commerce customer service, in *Proceedings of the 4th Workshop on e-Commerce and NLP* (2021), pp. 26–32
- A. Lipani, B. Carterette, E. Yilmaz, How am i doing?: Evaluating conversational search systems offline. *ACM Trans. Inf. Syst. (TOIS)* 39(4), 1–22 (2021)
- J. Lopes, How generic can dialogue breakdown detection be? the kth entry to dbdc3, in *Proceedings of Dialog System Technology Challenge 6* (2017)
- H. Lu, S. Bao, H. He, F. Wang, H. Wu, H. Wang, Towards boosting the open-domain chatbot with human feedback (2022). [arXiv:2208.14165](https://arxiv.org/abs/2208.14165)
- H. Mei, M. Bansal, M. Walter, Coherent dialogue with attention-based language models, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31 (2017)
- OpenAI, Gpt-4 technical report (2023). [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) [cs.CL]
- L. Ouyang, W. Jeffrey, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray et al., Training language models to follow instructions with human feedback, in *Advances in Neural Information Processing Systems* 35 (2022), pp. 27730–27744

- G. Pandey, D. Contractor, V. Kumar, S. Joshi, Exemplar encoder-decoder for neural conversation generation, in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018), pp. 1329–1338
- A. Paranjape, A. See, K. Kenealy, H. Li, A. Hardy, P. Qi, K.R. Sadagopan, N.M. Phu, D. Soylu, C.D. Manning, Neural generation meets real people: towards emotionally engaging mixed-initiative conversations, in *3rd Proceedings of Alexa Prize (Alexa Prize 2019)* (2020)
- A. Ram, R. Prasad, C. Khatri, A. Venkatesh, R. Gabriel, Q. Liu, J. Nunn, B. Hedayatnia, M. Cheng, A. Nagar, et al., Conversational ai: the science behind the alexa prize (2018)
- S. Roller, E. Dinan, N. Goyal, D. Ju, M. Williamson, Y. Liu, J. Xu, M. Ott, E.M. Smith, Y.-L. Boureau, et al., Recipes for building an open-domain chatbot, in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (2021), pp. 300–325
- H. Ryuichiro, K. Funakoshi, M. Inaba, Y. Tsunomori, T. Takahashi, K. Nobuhiro, Overview of dialogue breakdown detection challenge 3, in *Proceedings of Dialogue System Technology Challenge* (2017), p. 14
- T. Sandbank, M. Shmueli-Scheuer, J. Herzig, D. Konopnicki, J. Richards, D. Piorkowski, Detecting egregious conversations between customers and virtual agents, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)* (2018), pp. 1802–1811
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms (2017). [arXiv:1707.06347](https://arxiv.org/abs/1707.06347)
- A. See, C.D. Manning, Understanding and predicting user dissatisfaction in a neural generative chatbot, in *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2021), pp. 1–12
- I.V. Serban, A. Sordoni, Y. Bengio, A. Courville, J. Pineau, Building end-to-end dialogue systems using generative hierarchical neural network models, in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (2016), pp. 3776–3783
- I.V. Serban, R. Lowe, P. Henderson, L. Charlin, J. Pineau, A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse* **9**(1), 1–49 (2018)
- L. Shang, Z. Lu, 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* (2015), pp. 1577–1586
- X. Shen, H. Su, Y. Li, W. Li, S. Niu, Y. Zhao, A. Aizawa, G. Long, A conditional variational framework for dialog generation, in *55th Annual Meeting of the Association for Computational Linguistics, ACL 2017 (Association for Computational Linguistics (ACL), 2017)*, pp. 504–509
- X. Shen, H. Su, S. Niu, V. Demberg, Improving variational encoder-decoders in dialogue generation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018)
- W. Shi, E. Dinan, K. Shuster, J. Weston, J. Xu, When life gives you lemons, make cherryade: converting feedback from bad responses into good labels (2022). [arXiv:2210.15893](https://arxiv.org/abs/2210.15893)
- K. Shuster, M. Komeili, L. Adolphs, S. Roller, A. Szlam, J. Weston, Language models that seek for knowledge: modular search & generation for dialogue and prompt completion, in *Findings of the Association for Computational Linguistics: EMNLP 2022* (2022)
- K. Shuster, J. Urbanek, E. Dinan, A. Szlam, J. Weston, Dialogue in the wild: learning from a deployed role-playing game with humans and bots, in *Findings of the Association for Computational Linguistics: ACL-IJCNLP* (2021), pp. 611–624
- K. Shuster, J. Xu, M. Komeili, D. Ju, E.M. Smith, S. Roller, M. Ung, M. Chen, K. Arora, J. Lane, et al., Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage (2022). [arXiv:2208.03188](https://arxiv.org/abs/2208.03188)

- S. Steidl, C. Hacker, C. Ruff, A. Batliner, E. Nöth, J. Haas, Looking at the last two turns, i'd say this dialogue is doomed—measuring dialogue success, in *Text, Speech and Dialogue: 7th International Conference, TSD 2004, Brno, Czech Republic, September 8–11, 2004. Proceedings 7* (Springer, 2004), pp. 629–636
- N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, P.F. Christiano, Learning to summarize with human feedback, in *Advances in Neural Information Processing Systems 33* (2020), pp. 3008–3021
- Y. Sun, L. Wu, S. Song, X. Yu, X. He, G. Fu, Tracking satisfaction states for customer satisfaction prediction in e-commerce service chatbots, in *Proceedings of the 29th International Conference on Computational Linguistics* (2022), pp. 616–625
- A. Tamkin, M. Brundage, J. Clark, D. Ganguli, Understanding the capabilities, limitations, and societal impact of large language models (2021). [arXiv:2102.02503](https://arxiv.org/abs/2102.02503)
- J. Urbanek, A. Fan, S. Karamcheti, S. Jain, S. Humeau, E. Dinan, T. Rocktäschel, D. Kiela, A. Szlam, J. Weston, Learning to speak and act in a fantasy text adventure game, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP) (2019)* (2019), pp. 673–683
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in *Advances in Neural Information Processing Systems 30* (2017)
- A. Venkatesh, C. Khatri, A. Ram, F. Guo, R. Gabriel, A. Nagar, R. Prasad, M. Cheng, B. Hedayatnia, A. Metallinou, et al., On evaluating and comparing open domain dialog systems (2018). [arXiv:1801.03625](https://arxiv.org/abs/1801.03625)
- O. Vinyals, Q. Le, A neural conversational model (2015). [arXiv:1506.05869](https://arxiv.org/abs/1506.05869)
- M. Walker, D. Litman, C.A. Kamm, A. Abella, Paradise: a framework for evaluating spoken dialogue agents, in *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics* (1997), pp. 271–280
- M. Walker, R.J. Passonneau, J.E. Boland, Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems, in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics* (2001), pp. 515–522
- L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese, B. Balle, A. Kasirzadeh, et al., Ethical and social risks of harm from language models (2021). [arXiv:2112.04359](https://arxiv.org/abs/2112.04359)
- B.H. Wixom, P.A. Todd, A theoretical integration of user satisfaction and technology acceptance. *Inf. Syst. Res.* **16**(1), 85–102 (2005)
- Y. Wu, W. Wu, C. Xing, M. Zhou, Z. Li, Sequential matching network: a new architecture for multi-turn response selection in retrieval-based chatbots, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2017), pp. 496–505
- Y. Wu, W. Wu, D. Yang, C. Xu, Z. Li, Neural response generation with dynamic vocabularies, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018)
- C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, W.-Y. Ma, Topic aware neural response generation, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31 (2017)
- J. Xu, A. Szlam, J. Weston, Beyond goldfish memory: long-term open-domain conversation, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2022a), pp. 5180–5197
- J. Xu, M. Ung, M. Komeili, K. Arora, Y.-L. Boureau, J. Weston, Learning new skills after deployment: Improving open-domain internet-driven dialogue with human feedback (2022b). [arXiv:2208.03270](https://arxiv.org/abs/2208.03270)
- K. Yao, G. Zweig, B. Peng, Attention with intention for a neural network conversation model (2015). [arXiv:1510.08565](https://arxiv.org/abs/1510.08565)
- Y. Zhang, S. Sun, M. Galley, Y.-C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, W.B. Dolan, Dialogpt: large-scale generative pre-training for conversational response generation, in *Proceedings of the*

58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020b), pp. 270–278

W.X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al., A survey of large language models (2023). [arXiv:2303.18223](https://arxiv.org/abs/2303.18223)



Continual Learning for Task-Oriented Dialogue Systems

6

Task-oriented Dialogue Systems (ToDS) are widely popular now-a-days due to their pervasive usage in real-world applications like flight booking, customer service, virtual assistant services, etc. The main goal of these systems is to understand and complete tasks requested by users through multi-turn dialogues. Traditionally, ToDS are built with a number of subsystems, viz., Automatic Speech Recognition (ASR), Natural Language Understanding (NLU), Dialogue State Tracking (DST), Dialogue Policy (DP) Learning and Natural Language Generation (NLG), each playing its distinctive role in realizing the whole dialogue system (see Sect. 1.1 for more details).

Existing approaches to building these subsystems mainly rely on collecting a supervised dataset specific for each sub-task to be solved and then, training a model on this dataset to build the sub-module and finally, integrating them together to obtain the full system. There have been some recent approaches that attempted to solve all tasks jointly in an end to end manner. But, as these systems are build with offline training on a pre-collected domain specific dataset, they operate as *domain-specific* agents when deployed in practice. Thus, the system built for one domain (say, “Hotel”) does not naturally extend to other domains (say “flight booking”) and requires a whole system retraining incurring high costs.

Moreover, no matter how much supervised data is collected for a given domain, it is nearly impossible to ensure that the dialogue system will be able to handle all possible requests from the user with absolute accuracy. The inability to process user request by one subsystem often affects others down the pipeline due to error propagation. For example, considering the NLU sub-task, users may express some *unknown* intent or slots in their utterances which the NLU system may not recognize or miss-predict (during its post-deployment use) due to its limitation of knowledge obtained through offline pre-deployment training. This will affect the subsequent component like DST, DP and NLG giving rise to undesirable or irrelevant response from the dialogue system and subsequently, incompleteness of the task requested by

the user. Identifying such open intents in user utterances can reduce false-positive errors and improve customer satisfaction.

Building continual learning systems for task-oriented dialogues that can scale over domains and deal with unknown requests is thus essential for the success of ToDS in practice. In this chapter, we present and discuss the recent progresses made on this topic. There have been a number of works (Shen et al. 2019, Madotto et al. 2021; Liu et al. 2021a; Zhu et al. 2022; Mi et al. 2020; Zhao et al. 2022; Zhang et al. 2022; Yang et al. 2022) that have addressed the problem of continual learning on ToDS. We discuss some of the representative ones here. Section 6.1 discusses and presents methods for open world intent detection and learning. In Sect. 6.2, we discuss continual learning for the semantic slot filling task. Section 6.3 covers the recent progress on continual learning for dialogue state tracking and Sect. 6.4 presents the works on continual learning for natural language generation. We also discuss recent approaches that attempted to solve all ToDS sub-tasks jointly in end-to-end manner and in continual learning setting over multiple domains (Sect. 6.5). Finally, we conclude with a chapter summary in Sect. 6.6.

6.1 Open Intent Detection and Learning

Open Intent Detection (OID) is a challenging problem in Spoken Language Understanding (SLU). Unlike traditional intent detection methods (Jurafsky and Martin 2020; Zhang et al. 2019; Haihong et al. 2019; Wang et al. 2018b; Goo et al. 2018) that is concerned with classifying an user utterance into one of the *pre-defined* and *known* intent classes, OID needs to identify open (i.e., unknown) intents during testing, besides the known ones. Figure 6.1 shows examples for the OID task. Here, for the first four utterances, the corresponding intent label (class) is known (exists in the training set). Whereas, for the last two utterances, the intent label is unknown or open. Detecting open intents are important not only because it can

User utterances	Intent Label
Book a flight from LA to Madrid.	Book flight
Can you get me a table at Steve's?	Restaurant reservation
Book Delta ticket Madison to Atlanta.	Book flight
Schedule me a table at Red Lobster.	Restaurant reservation
...	...
Can you tell me the name of this song?	Open
Look up the calories in an apple.	Open

Fig. 6.1 Example of open intent classification (Source Zhang et al. 2021a)

tell the user that the system cannot handle the unknown intents, but is also the pre-condition or motivation for post-deployment continual learning of new intents. Please refer to Chap. 2 for details, where open intents are called novelties.

Formally, the OID task can be defined as an $(n + 1)$ -class classification problem (Shu et al. 2017; Lin and Xu 2019a; Fei and Liu 2016), where the goal is to classify the n -class known intents into their corresponding classes correctly while identifying the $(n + 1)$ th class of open intent. There have been a number of works (Lin and Xu 2019a; Yan et al. 2020; Lin et al. 2020; Lin and Xu 2019b; Zhan et al. 2021; Zhang et al. 2021a, b) that have addressed this problem in recent years. We discuss two representative approaches in details in the following subsections.

6.1.1 Open Intent Detection with Margin Loss

Lin and Xu (2019a) have proposed a two-stage method for open intent detection. The approach uses a BiLSTM layer (Mesnil et al. 2014) with the margin loss as the feature extractor. In particular, Large Margin Cosine Loss (LMCL) (Wang et al. 2018a) is used to learn discriminative deep features by forcing the network to *maximize inter-class variance* and *minimize intra-class variance*. Then, the features are fed to the density-based novelty detection algorithm, called the local outlier factor (LOF) (Breunig et al. 2000) for novelty detection. Figure 6.2 shows the architecture of the model.

Representation Learning with Large Margin Cosine Loss (LMCL). Given a user utterance $w_{1:\ell}$ with maximum word sequence length ℓ , the model transforms the sequence into word embeddings and feeds them to a BiLSM layer. The last output vector of forward

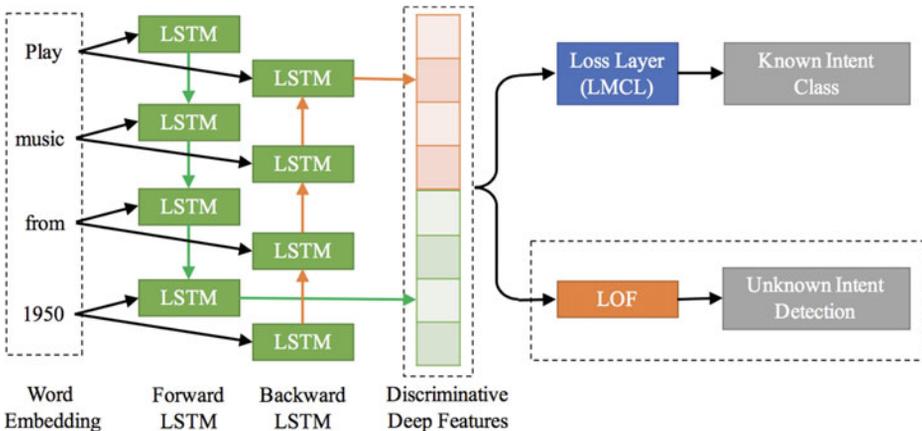


Fig. 6.2 Open intent classification model based on LMCL loss and local outlier factor (LOF) (Source Lin and Xu 2019a)

LSTM and the first output vector of backward LSTM are then concatenated to generate the final utterance representation as x which is used as the input of the next stage.

To learn the representation x , the commonly used softmax loss of BiLSTM is replaced with LMCL loss, defined as follows:

$$\mathcal{L}_{LMC} = \frac{1}{N} \sum_i -\log \frac{e^{s \cdot (\cos(\theta_{y_i, i}) - m)}}{e^{s \cdot (\cos(\theta_{y_i, i}) - m)} + \sum_{j \neq y_i} e^{s \cdot \cos \theta_{j, i}}}, \quad (6.1)$$

constrained by

$$\begin{aligned} \cos(\theta_j, i) &= W_j^T x_i, \\ W &= \frac{W^*}{\|W^*\|}, \quad x = \frac{x^*}{\|x^*\|}, \end{aligned} \quad (6.2)$$

where N denotes the number of training samples, y_i is the ground-truth class label of the i -th sample, s is the scaling factor, m is the cosine margin, W_j is the weight vector of the j -th class, and θ_j is the angle between W_j and x_i .

LMCL transforms softmax loss into cosine loss by applying L2 normalization on both features and weight vectors. It further maximizes the decision margin in the angular space. With normalization and cosine margin, LMCL forces the model to maximize inter-class variance and to minimize intra-class variance.

Novelty Detection with Local Outlier Factor (LOF). In the second stage, the deep discriminative features x is fed to the LOF algorithm to help detect unknown intents in the context of local density. LOF is computed as follows:

$$\text{LOF}_k(A) = \frac{\sum_{B \in N_k(A)} \frac{\text{lrd}(B)}{\text{lrd}(A)}}{|N_k(A)|} \quad (6.3)$$

where $N_k(A)$ denotes the set of k -nearest neighbors and lrd denotes the local reachability density defined as:

$$\text{lrd}_k(A) = \frac{|N_k(A)|}{\sum_{B \in N_k(A)} \text{reachdist}_k(A, B)} \quad (6.4)$$

where $\text{lrd}_k(A)$ denotes the inverse of the average reachability distance between object A and its neighbors. Here, $\text{reachdist}_k(A, B)$ is defined as the following:

$$\text{reachdist}_k(A, B) = \max\{k\text{-dist}(B), d(A, B)\} \quad (6.5)$$

where $d(A, B)$ denotes the distance between A and B , and $k\text{-dist}$ denotes the distance of the object A to the k th nearest neighbor.

For open intent detection, if an example's local density is significantly lower than that of its k -nearest neighbors, the model infers the example to be of an unknown intent.

6.1.2 Learning Adaptive Decision Boundary

Finding appropriate decision boundary to separate known and unknown classes is crucial for OID task performance. Most of the existing methods (Bendale and Boulton 2015; Shu et al. 2017; Lin and Xu 2019a; Fei and Liu 2016) design specific classifiers for identifying open intents. The performance often suffers due to lack of learning optimal decision conditions while learning the classification task itself. To deal with this problem, Zhang et al. (2021a) have proposed a post-processing method to learn the adaptive decision boundary (ADB) to solve the OID task.

As shown in Fig. 6.3, the model first extracts the intent representations from the BERT model (Devlin et al. 2019). Then, it pre-trains the model under the supervision of the softmax loss. The model defines centroids for each known class and assume that the known intent features fall into in the closed ball area formed by their corresponding decision boundaries (the radius of each ball area). Thus, to obtain the decision boundary, the model aims to learn the radius of each ball area. The suitable decision boundary for each class should satisfy two constraints:

- (1) They should be broad enough to surround all in-domain samples as much as possible.
- (2) they need to be tight enough to prevent out-of-domain samples from being identified as in-domain samples.

To learn the decision boundary for each known class, the model initializes the boundary parameters with standard normal distribution and use a learnable activation function to project them into the positive radius of each decision boundary. For this purpose, a loss function was proposed which optimizes the boundary parameters by balancing both the

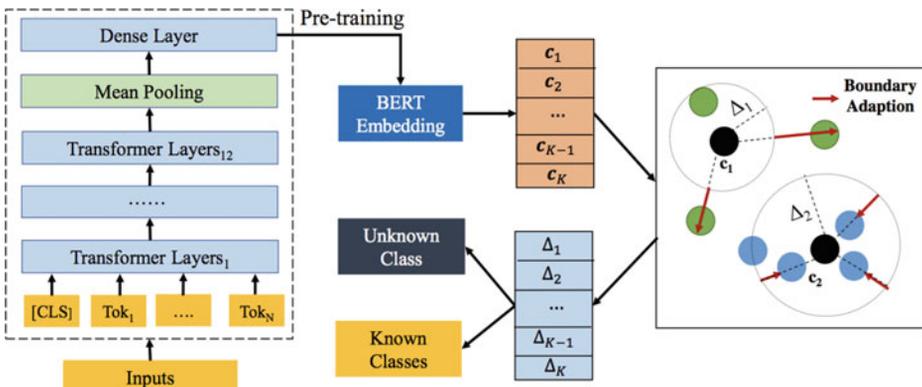


Fig. 6.3 Model Architecture for learning Adaptive Decision Boundary (Source Zhang et al. 2021a)

open space risk and the empirical risk (Scheirer et al. 2013). The learned adaptive decision boundaries for the known classes are then used for open intent classification.

ADB Model Overview. The ABD model consists of two main modules. The first module, Intent Representation Learning, aims to learn intent features from input user utterance using the BERT model. The second module, Decision Boundary learning, aims to learn adaptive decision boundary for each of the known classes and use it for open intent classification.

Intent Representation Learning. Given the i th input sentence (user utterance) s_i , the representation learning layer first gets all its token embeddings $[CLS, T_1, \dots, T_N] \in \mathbb{R}^{(N+1) \times H}$ from the last hidden layer of BERT. Then, it performs mean-pooling on these token embeddings to get the semantic embedding \mathbf{x}_i of s_i as follows:

$$\mathbf{x}_i = \text{mean-pooling}([CLS, T_1, \dots, T_N]), \quad (6.6)$$

where CLS is the vector for text classification in BERT, N is the sequence length and H is the hidden layer size. Next, \mathbf{x}_i is fed to a dense layer h to get the intent representation $\mathbf{z}_i \in \mathbb{R}^D$:

$$\mathbf{z}_i = h(\mathbf{x}_i) = \sigma(W_h \mathbf{x}_i + b_h), \quad (6.7)$$

where σ is a ReLU activation function and W_h, b_h denote the weights and the bias of layer h .

As the decision boundaries learn to adapt to the intent feature space, the model needs to learn the intent representation first. Due to the lack of open intent samples, the model uses known intents as prior knowledge to pre-train the model. The intent feature \mathbf{z}_i is learned by minimizing the softmax loss. This pre-trained model is used to extract intent features for learning decision boundaries.

Decision Boundary Learning. Let $S = \{(\mathbf{z}_1, y_1), \dots, (\mathbf{z}_N, y_N)\}$ be the known intent embeddings of examples with their corresponding labels and S_k denotes the embeddings set of examples for class- k . The centroid $\mathbf{c}_k \in \mathbb{R}^D$ is the mean vector of the embeddings in S_k :

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{z}_i, y_i) \in S_k} \mathbf{z}_i \quad (6.8)$$

where $|S_k|$ denotes the size of or the number of embeddings in S_k .

Also, let the radius of the decision boundary of class- k with respect to the centroid \mathbf{c}_k be Δ_k . Then, for each known intent embedding $\mathbf{z}_i \in S_k$, the model needs to satisfy the following constraint:

$$\forall \mathbf{z}_i \in S_k, \|\mathbf{z}_i - \mathbf{c}_k\|_2 \leq \Delta_k, \quad (6.9)$$

where $\|\mathbf{z}_i - \mathbf{c}_k\|_2$ denotes the Euclidean distance between \mathbf{z}_i and \mathbf{c}_k . That is, we hope that examples belonging to class- k are constrained in the ball area with centroid \mathbf{c}_k and radius Δ_k .

Boundary Learning. The decision boundaries should be adaptive to the intent feature space to balance both empirical and open space risk (Bendale and Boulton 2015). Note, if $\|\mathbf{z}_i - \mathbf{c}_{y_i}\|_2 > \Delta_{y_i}$, it indicates that the known intent samples fall outside their corresponding decision boundaries, which increases the empirical risk. On the other hand, if $\|\mathbf{z}_i - \mathbf{c}_{y_i}\|_2 < \Delta_{y_i}$ and Δ_{y_i} is broader, though more known intent samples are likely to fall within the broader decision boundaries, it may also introduce more open intent samples inside the boundary, increasing the open space risk.

To balance both risks, the model optimizes a boundary loss \mathcal{L}_b for learning the boundary parameters Δ_{y_i} as follows:

$$\mathcal{L}_b = \frac{1}{N} \sum_{i=1}^N [\delta_i (\|\mathbf{z}_i - \mathbf{c}_{y_i}\|_2 - \Delta_{y_i}) + (1 - \delta_i) (\Delta_{y_i} - \|\mathbf{z}_i - \mathbf{c}_{y_i}\|_2)], \quad (6.10)$$

where y_i is the label of the i th sample and δ_i is defined as:

$$\delta_i := \begin{cases} 1, & \text{if } \|\mathbf{z}_i - \mathbf{c}_{y_i}\|_2 > \Delta_{y_i}, \\ 0, & \text{if } \|\mathbf{z}_i - \mathbf{c}_{y_i}\|_2 \leq \Delta_{y_i}. \end{cases} \quad (6.11)$$

Open Classification with ADB. After training, the centroid and the learned decision boundary of each known class k are used in inference as follows:

$$\hat{y} = \begin{cases} \text{open, if } d(\mathbf{z}_i, \mathbf{c}_k) > \Delta_k, \forall k \in \mathcal{Y}; \\ \arg \min_{k \in \mathcal{Y}} d(\mathbf{z}_i, \mathbf{c}_k), \text{ otherwise,} \end{cases} \quad (6.12)$$

where $d(\mathbf{z}_i, \mathbf{c}_k)$ denotes the Euclidean distance between \mathbf{z}_i and \mathbf{c}_k , and $\mathcal{Y} = \{1, 2, \dots, K\}$ denotes the set of known intent labels.

6.2 Continual Learning for Semantic Slot Filling

Semantic slot filling is one of the major tasks in spoken language understanding (SLU) that aims to extract semantic slots and their values that the user wants the system to understand from his/her utterance with respect to his/her intent (Jurafsky and Martin 2020). Formally, the task is defined as a sequential labeling problem where, given a natural language utterance x , the goal is to map x into a slot label sequence y of the same length in the IOB format (Yao et al. 2014). Existing works on slot filling (Yao et al. 2014; Peng et al. 2015; Kurata et al. 2016; Hakkani-Tür et al. 2016; Liu and Lane 2016; Deng et al. 2019; Ray et al. 2019; Guo et al. 2014; Zhang and Wang 2016; Mesnil et al. 2014; Goo et al. 2018) are mainly trained offline on large scale corpora with pre-collected utterances. Thus, when deployed, they cannot cover all possible varieties of utterances exhaustively (e.g., personalized expressions, new

vocabulary, utterances with new intents) and the performance drops. Continual learning of new semantic slots thus is crucial for the success of SLU.

Motivated by progressive neural networks (Rusu et al. 2016), Shen et al. (2019) have recently designed a progressive slot filling model (ProgModel) to enable continual learning for slot filling. The goal is to keep learning new utterances efficiently while retaining old knowledge without any need for retraining on old training data. ProgModel consists of a context gate that transfers previously learned knowledge to a small-sized expanded component. Using the transferred knowledge, each newly expanded component is then trained in a progressive manner with new data. The previously trained components remains untouched to avoid catastrophic forgetting.

ProgModel: Architecture. As shown in Fig. 6.4, ProgModel consists of the following two components:

Expanded Component (M_t). At each batch t , a new component M_t [see Fig. 6.4] is expanded on the base model M_0 and previously expanded components M_1, \dots, M_{t-1} , denoted as $M_{<t}$. Each component is built based on Att-BiRNN model (Liu and Lane 2016). M_t is learned from scratch at each batch t such that it can learn the new sentence structures via word sequence correlations. Additionally, two knowledge transfer mechanisms (green parts in Fig. 6.4) are employed to maximally leverage the previously learned knowledge in $M_{<t}$ while building M_t as follows:

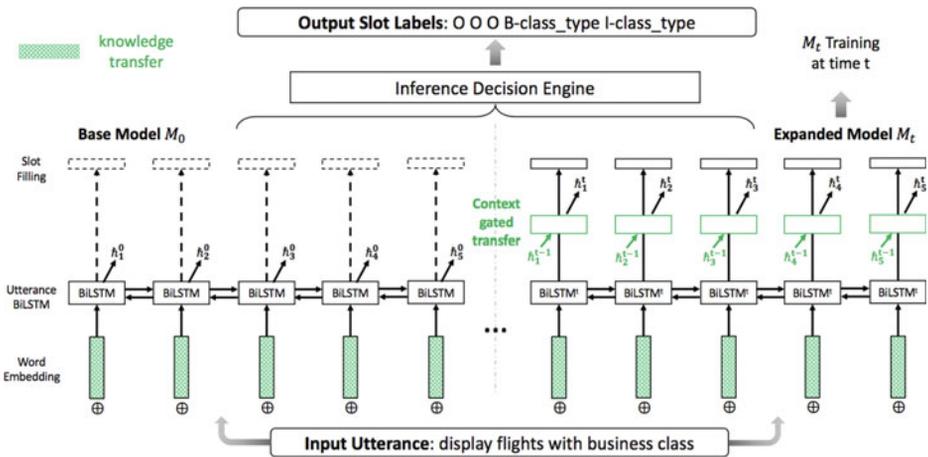


Fig. 6.4 ProgModel Architecture (Source Shen et al. 2019)

1. *Word Embeddings Transfer*: Word embeddings in the newly expanded component M_t are initialized using those in M_0 based on the assumption that M_0 covers most of the vocabulary. For a new word w , its word embedding in M_t is initialized using GloVe embedding. During training of M_t , the embeddings are all fine-tuned.
2. *Gated Utterance Context Transfer*: To avoid retraining on all data or repeatedly duplicating a large base model every time, a context gate is designed by introducing the lateral connections from the most recent component M_{t-1} . Specifically, the lateral connections transfer the hidden states \tilde{h}_i^{t-1} for each word i from M_{t-1} to the newly expanded component M_t . They are projected to the same dimensional space of \tilde{h}_i^t via projection matrix V^t shared for each word as follows.

$$\tilde{h}_i^t = h_i^t + \text{sigmoid}(V^t \cdot \tilde{h}_i^{t-1}) \quad (6.13)$$

Then, the context vector c_i^t for the i th word in M_t is computed as:

$$c_i^t = \sum_{k=1}^n \alpha_{i,k}^t \tilde{h}_k^t \quad (6.14)$$

where $\alpha_i^t = \text{softmax}(e_i)$ is fine-tuned based on α_i^0 trained in base model M^0 following the Att-BiRNN model (Liu and Lane 2016). Here, $\alpha_{i,j}^0$ is computed as:

$$\alpha_{i,j}^0 = \frac{\exp(e_{i,j}^0)}{\sum_{k=1}^T \exp(e_{i,k}^0)}, \quad (6.15)$$

where $e_{i,k}^0 = g^0(h_{i-1}^0 \oplus c_{i-1}^0, h_k^0)$ is learned from a feed forward neural network g^0 . In ProgModel, each model M^t has an independent g^t which is initialized by g^0 in M^0 .

Inference Decision Engine. The inference decision engine (IDE) derives the slot filling label output. It is built as a non-trainable separate component to avoid the potential catastrophic forgetting. Two types of decision engines were proposed:

- (1) t-IDE: Here, ProgModel uses only the output of M^t as the decision engine.
- (2) c-IDE: For this case, for i th word, ProgModel combines all outputs from each component M^t , as $\sum_{k=0}^t P^k(i) I^k(i)$, where $I^k(i)$ is an indicator function which is 1 when the i th word is in the vocabulary of M^k and 0 otherwise. The label with maximum probability is then selected for inference.

Shen et al. (2019) also empirically showed that c-IDE outperforms t-IDE and other baselines in terms of slot F1 score.

6.3 Continual Learning for Dialogue State Tracking

Given a user utterance and the dialogue history, Dialogue State Tracking (DST) (Williams et al. 2014; Henderson et al. 2014; Mrkšić et al. 2017; Shan et al. 2020) aims to output an accurate dialogue state, often represented as a set of slot-value pairs, such as $\{(\text{CUSINE}, \textit{italian}), (\text{PRICE_RANGE}, \text{\$\$})\}$. Most DST models are usually trained offline, which requires a fixed dataset prepared in advance. These offline solutions are often impractical in real-world applications, as online dialogue systems usually involve continually emerging new data and domains as new services are introduced. It is also infeasible to retrain the DST models from scratch every time when a new domain data arrives due to computational costs, storage budgets and data privacy. We discuss two recent representative works that address this problem and proposed continual learning solution below.

6.3.1 Domain-Lifelong Learning for DST

Liu et al. (2021a) have explored the problem of Domain-Lifelong Learning for DST (DLL-DST), which aims to continually train a DST model on new domains while avoiding catastrophically forgetting the old ones. The DST task here is formulated as a slot-filling task—at each dialogue turn, the DST model takes the user utterance and the dialogue history as input and predicts the value for each slot. The DLL-DST task continually trains DST model on new data (to learn new domains) that arrives in a stream form $\{\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_N\}$, where, at each step, the new data \mathcal{D}_i can contain one or multiple new domains.

Existing mainstream lifelong learning (LL) (or continual learning) techniques such as *regularization-based* Kirkpatrick et al. (2017) and *replay-based* Rusu et al. (2016); Lopez-Paz and Ranzato 2017 methods (also see Chap. 2) can be applied to solve the DLL-DST problem. However, when deploying existing replay-based methods on the DLL-DST task, Liu et al. (2021a) have found that they suffer from two major drawbacks:

1. **Expression diversity:** In the DST task, dialogue texts usually contain a variety of expressions for each dialogue state, The expression diversity makes it difficult for previous replay-based methods to select the most representative old samples.
2. **Combinatorial explosion:** The classes of dialogue states can explode rapidly as the number of slot-value pairs increases, which makes it hard for the model to reserve at least one sample for each dialogue state in old domains.

Thus, to solve DLL-DST effectively, a domain-lifelong learning method called *Knowledge Preservation Networks* (KPN) is designed. KPN is model-agnostic by design and can be used with many DST models. Liu et al. (2021a) have adopted the SOM-DST model (Kim et al. 2020c) for the study. At each dialogue turn, SOM-DST considers the last system response and the last dialogue state as the dialogue history and combines them with the current user

utterance into an input sequence for the BERT encoder (Devlin et al. 2019). Besides the [CLS] and [SEP] tokens, a special token [SLOT] is placed at the beginning of each slot in the last dialogue state. The BERT encoder obtains the contextual representation for the input sequence. The encoded hidden state of [SLOT] is used as the feature vector of each slot.

Knowledge Preservation Networks (KPN). Given a model trained on the old data, KPN aims to continually learn a unified DST model for all observed domains so far based on a new combined dataset ($D_i^{train} \cup \mathcal{P}$). Here, D_i^{train} is the training data of the new domains at step i and \mathcal{P} is a *bounded* memory that stores *representative* old samples, denoted as $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \dots, \mathcal{P}_m\}$. Here, \mathcal{P}_i is the set of stored samples for the i th domain and m is the number of old domains. KPN consists of two main components that deal with problems (1) and (2) as follows:

Multi-prototype enhanced retrospection. To deal with problem (1), KPN uses multi-prototype enhanced retrospection, which computes multiple slot prototypes for each domain and selects the most representative old samples based on these slot prototypes. The process works as follows. After learning the new domains, KPN stores $|\mathcal{P}_i| = \frac{B}{m}$ samples for each new domain, where m is the number of all observed domains and B is the total number of samples that can be saved. All samples of the i th domain are encoded into the hidden representation and a prototype μ^s for each slot s in this domain is compute as:

$$\mu^s = \frac{1}{|N|} \sum_{x \in N} f^s(x) \quad (6.16)$$

where N is the training samples of the i th domain and $f^s(x)$ is the hidden state of [SLOT] in front of the slot s , which is the slot representation.

Next, the distance between the slot representation of each training sample and the corresponding slot prototype is computed. Based on the average distance of all slots, a sorted list of new training samples is produced. Then, the top $\frac{B}{m}$ samples are selected from this list as exemplars to be stored in the memory.

Multi-strategy knowledge distillation. To cope with problem (2), KPN employs multi-strategy knowledge distillation. The idea is to preserve the knowledge of the model trained in the last step through multiple distillation strategies. Four types of distillation strategies, viz., Encoder Feature Distillation, Classifier Prediction Distillation, Decoder Feature Distillation and Generation Prediction Distillation are used for this purpose. During each step of the domain-lifelong learning process, KPN combines these four types of distillation losses to train the DST model. At each step, the training set ($D_i^{train} \cup \mathcal{P}$) is used to train the model with the combined loss and then, the most representative samples are selected to update the memory. We refer the reader to Liu et al. (2021a) for further details about the distillation losses in KPN.

Figure 6.5 shows an example of continual learning involving three domains in KPN. When learning the *Flight* domain, the model is updated with the combination of training data for the *Flight* domain and the old data stored in the replay memory. The numbers 1 to 4 here

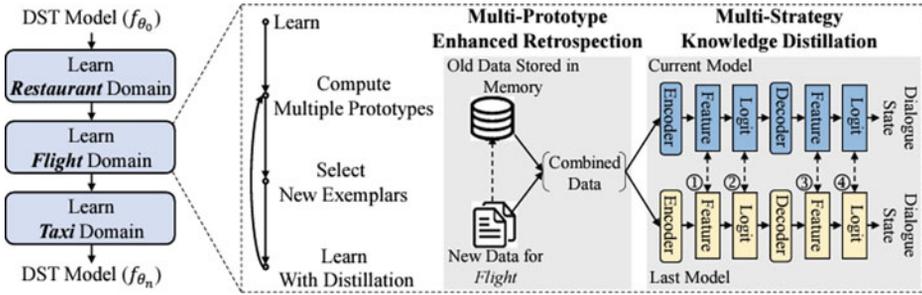


Fig. 6.5 An example of continually learning three domains in KPN (Source Liu et al. 2021a)

represent four different knowledge distillation strategies used to retain previous knowledge. Then, the method selects new representative samples to reserve via multi-prototype enhanced retrospection.

6.3.2 Continual Prompt Tuning for DST

Zhu et al. (2022) have proposed a parameter-efficient framework called Continual Prompt Tuning (CPT) for DST. CPT is based on prompt tuning (Lester et al. 2021) and it not only avoids forgetting but also enables knowledge transfer between tasks. As illustrated by the yellow components in Fig. 6.6, CPT concatenates the input with a few *tunable* task-specific prompt tokens before feeding it to a *frozen* pre-trained model. Since these prompt tokens have only a small number of parameters (0.1% of the pre-trained model’s parameters in their experiments), the model can be trained and the prompt for each task can be stored efficiently. During inference, the same pre-trained model can deal with different tasks by inputting different prompts.

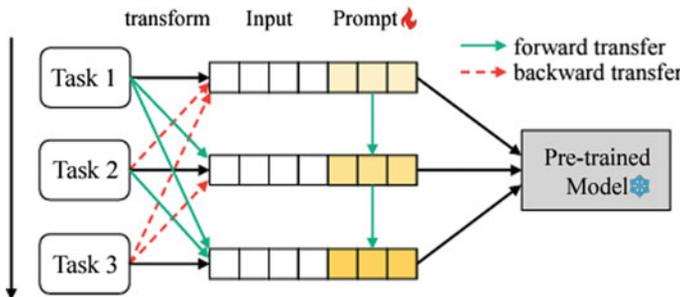


Fig. 6.6 An illustration of Continual Prompt Tuning (Source Zhu et al. 2022)

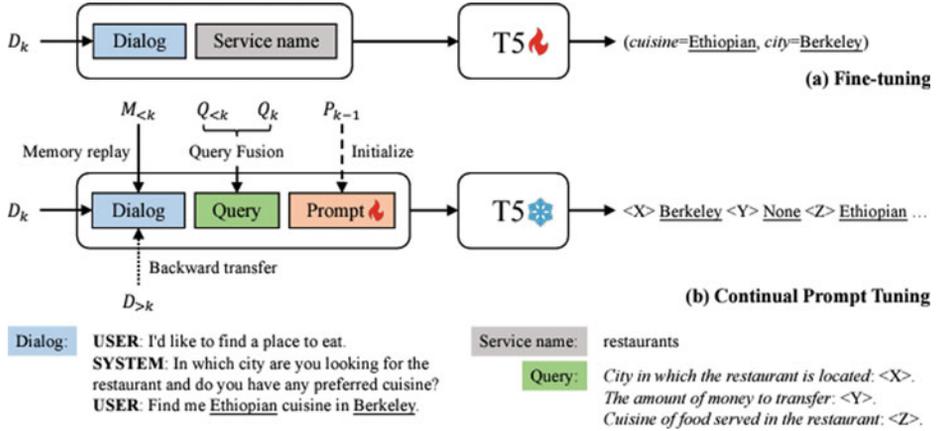


Fig. 6.7 An illustration of fine-tuning and continual prompt tuning for continual DST (Source Zhu et al. 2022)

In CPT, each service/API is treated as a task (here, service and task are used interchangeably). CPT continually learns new services that have multiple slots to fill. Recently, Lee et al. (2021), Lin et al. (2021a) have shown that incorporating a natural language description for each slot is beneficial for DST. Following this idea, CPT also concatenates all slots' descriptions with the input and a sentinel token is inserted after each description, formulating DST as a *masked spans recovering task* that generates all slot values in one pass.

Figure 6.7 shows an illustration of fine-tuning and continual prompt tuning for continual DST. The *Fine-tuning* approach takes the dialogue and the current service name as input and tunes T5 model to generate slot-value pairs. Instead of fine-tuning a pre-trained model as shown in Fig. 6.7a, CPT (Fig. 6.7b) feeds the *dialogue*, *query* consisting of slot descriptions and sentinel tokens, and *prompt tokens* as inputs to the *frozen* T5 model and tunes the prompt's embeddings to generate values for all slots in the query. Figure 6.7 also shows an example dialogue, service name, fused query, and expected outputs. Slot *names* and *descriptions* are in italic and values are underlined. Here, the second slot description in the query belongs to another service (“banks”) and is inserted by query fusion.

Different from the vanilla approach of training each task's prompt from scratch and fixing it afterward, CPT enables *knowledge transfer* from both preceding tasks (forward transfer) as well as subsequent tasks (backward transfer). To realize forward transfer (green solid arrows in Fig. 6.6), CPT employs several techniques such as continual prompt initialization, query fusion and memory replay. For (positive) backward transfer (red dashed arrows in Fig. 6.6), CPT uses a memory-guided technique that uses subsequent tasks' data to update the previous tasks' prompts selectively.

6.4 Continual Learning for Natural Language Generation

Natural language generation (NLG) aims to produce a natural language utterance containing the desired information given a dialogue act (see Chap. 1 for more details). Existing NLG models are typically trained offline using annotated data from a single or a fixed set of domains. However, an NLG model should be able to continually learn new utterance patterns without forgetting the old ones it has already learned. Below, we discuss two recent approaches that have studied NLG in continual learning setting to improve the response generation capability over time.

6.4.1 Adaptively Regularized Prioritized Exemplar Replay (ARPER)

Let \mathbf{d} denotes a dialogue act, defined as the combination of *intent* and a set of *slot-value pairs*. Then, given \mathbf{d} , a NLG model f_θ generates an utterance containing the desired information in \mathbf{d} . Given a ground truth utterance $\mathbf{Y} = (y_1, y_2, \dots, y_K)$ corresponding to the dialogue act \mathbf{d} , the model f_θ is usually trained by minimizing the average cross-entropy loss L_{CE} with respect to all tokens in \mathbf{Y} .

Mi et al. (2020) have studied the aforementioned NLG problem in a continual learning (CL) setting, where the goal is to expand the NLG model’s knowledge to new domains incrementally. The problem setting assumes that the new data arrive task by task. In a new task t , the new data D_t is used to train the NLG model $f_{\theta_{t-1}}$ obtained till the last task and the updated model f_{θ_t} (after training with D_t) needs to perform well on all tasks learned so far. To this end, a method called *Adaptively Regularized Prioritized Exemplar Replay* (ARPER) is proposed to effectively deal with catastrophic forgetting in CL. The approach is discussed below.

Prioritized Exemplar Replay. To prevent the NLG model catastrophically forgetting utterance patterns in earlier tasks, ARPER stores a small set of representative utterances from previous data, namely *exemplars* and replays them to the NLG model each time it needs to be trained on the new task data. For this, a prioritized exemplar selection scheme is designed to choose the representative and diverse exemplars for NLG. The following two criteria are used here:

(1) **Representative utterances.** The replay Exemplars \mathbf{E}_t of a task t should be representative of dataset \mathbf{D}_t . \mathbf{E}_t is selected as a *priority list* from \mathbf{D}_t that *minimizes* a priority score:

$$U(\mathbf{Y}, \mathbf{d}) = L_{CE}(\mathbf{Y}, \mathbf{d}, \theta_t) \cdot |S(\mathbf{d})|^\beta, \quad (6.17)$$

where $S(\mathbf{d})$ is the set of slots in \mathbf{Y} and β is a hyper-parameter. This formula correlates the representativeness of an utterance to its L_{CE} . The second term $|S(\mathbf{d})|^\beta$ controls the importance of the number of slots in an utterance to be prioritized as exemplars.

(2) **Diverse utterances.** Exemplars should contain diverse slots of the task rather than being similar or repetitive. A drawback of score $U(\cdot)$ is that similar or duplicated utterances

containing the same set of frequent slots could be prioritized over that with respect to a diverse set of slots. To encourage diversity, an iterative approach is used to add data from \mathbf{D}_t to the priority list \mathbf{E}_t while using $U(\cdot)$. In each iteration, if the set of slots of the current utterance is already covered by utterances in \mathbf{E}_t , the model skips it and moves on to the data with the next best priority score.

Reducing Exemplars in Previous Tasks. ARPER stores a *fixed* number of exemplars throughout the entire continual learning process to maintain a bounded memory footprint as in (Rebuffi et al. 2017). As more tasks are continually learned, exemplars in previous tasks are gradually reduced. This is achieved by only keeping the ones in the *front* of the priority list and the exemplar size of a task is set to be proportional to the training data size of the task for differentiating the task’s difficulty.

Constraint with Adaptive Elastic Weight Consolidation. Although exemplars of previous tasks are stored and replayed, the size of exemplars should be reasonably small to reduce memory overhead. To enforce a stronger constraint, a regularization-based method Elastic Weight Consolidation (EWC) (Kirkpatrick et al. 2017) is employed. EWC utilizes a quadratic term to elastically regularize parameters important for previous tasks. A scalar λ is used to control the contribution of the quadratic regularization term.

As new tasks can have different difficulties and similarities compared to previous ones, the degree of need to preserve the previous knowledge varies. Thus, an adaptive weight (λ) for the EWC regularization term is introduced as:

$$\lambda = \lambda_{base} \sqrt{V_{1:t-1}/V_t}, \quad (6.18)$$

where $V_{1:t-1}$ is the *old* word vocabulary size in previous tasks, and V_t is the *new* word vocabulary size in the current task t ; λ_{base} is a hyper-parameter. Here, λ increases when the ratio of the size of old word vocabularies to that of new ones increases, making the regularization term more important when the new task contains fewer new vocabularies to learn.

6.4.2 Continual Learning with Transformer Calibration

Yang et al. (2022) have proposed a transformer-based generation model for NLG that attempts to mitigate catastrophic forgetting in online continual learning (CL) using attention calibration. The problem of online CL is defined as a sequence of training tasks in CL such that the data of each task come and go in an online or streaming fashion. Thus, effectively the system can only train on the data of each task in one epoch Guo et al. (2022b).

To solve the problem, an attention calibration technique is introduced for transformer-based NLG models. The goal is to seek a better balance between stability and plasticity of CL algorithms via neuron calibration. Here, attention in the transformer is modeled as a calibrated unit that is independent from the pre-trained base model in order to preserve the

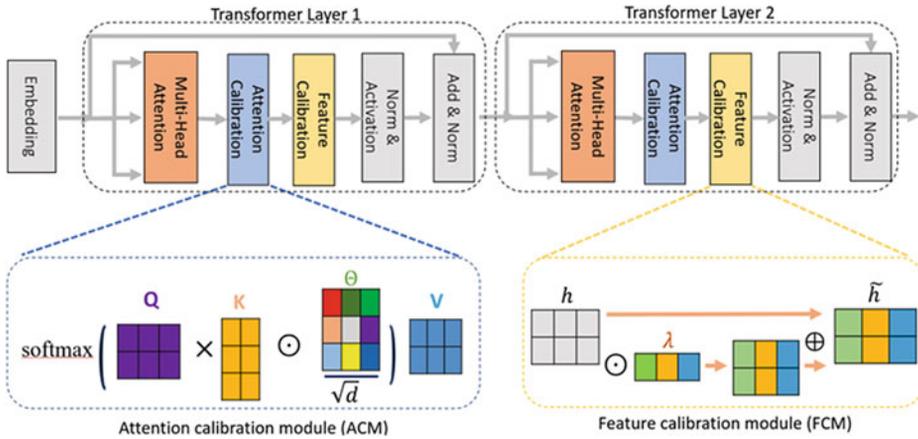


Fig. 6.8 Transformer calibration for continual learning for NLG (Source Yang et al. 2022)

learned knowledge and to avoid catastrophic forgetting. Figure 6.8 provides an illustration of the neuron calibration process.

Formally, two types of general calibration modules have been designed in the calibrated transformer model: (1) *attention calibration module* (ACM) and (2) *feature calibration module* (FCM). ACM learns to scale the attentions of the transformer function whereas FCM learns to scale the feature map output from the transformer block. These modules are sequentially applied to the layers in the multi-head attention model (as shown in the figure) to calibrate the attention signals and feature maps respectively. The calibrated transformer model is trained based on elastic weight consolidation (EWC) (Kirkpatrick et al. 2017) to mitigate the catastrophic forgetting problem.

6.5 Joint Continual Learning of All Dialogue Tasks

Building a full fledged task-oriented dialogue system (ToDS) involves solving multiple dialogue tasks (see Chap. 1 for more details). These tasks can be learned independently and combined in a pipelined fashion or can be learned jointly end to end. Jointly learning these tasks in a continual manner can improve the performance of the system holistically. Below, we discuss two representative recent approaches along this direction. The first work (Madotto et al. 2021) introduces the end-to-end continual learning (CL) problem for ToDS and designs a model called AdapterCL to solve it. We cover this work in more details here. The second one (Zhang et al. 2022) introduces a self-learning framework SL-AGENT for building end-to-end ToDS. We briefly discuss it at the end of this section.

6.5.1 End-to-End Continual Learning for ToDS with AdapterCL

Madotto et al. (2021) have recently proposed a number of continual learning (CL) benchmarks for ToDS with 37 domains to be learned continually on four settings: *intent recognition* (INTENT), *dialogue state tracking* (DST), *natural language generation* (NLG) and *end-to-end* (E2E). The goal is to train a model on data from a set of domains sequentially (e.g., first on the hotel domain D_{Hotel} and then on the Taxi D_{Taxi} and so on) as shown in Fig. 6.9. The model parameter is updated sequentially based on the loss function L .

Tasks formulation. ToDS in (Madotto et al. 2021) is modeled as a seq2seq generation task (Lei et al. 2018; Byrne et al. 2021) that generates both API-calls and responses. As shown in Fig. 6.10, the model uses the dialogue history to generate an API-call, which is the concatenation of the user intent plus the current dialogue state and uses its output (which can be empty or the system speech-act) to generate the system response. This modelling choice is guided by the existing annotated dialogue datasets, which provide the intent and the dialogue state of the user at every turn and the speech-act of the system and allows to define the aforementioned four settings of ToDS.

Let the dialogue history be H which is a single sequence of tokens from the concatenation of the alternating utterances from the user and the system turns respectively. H has all the

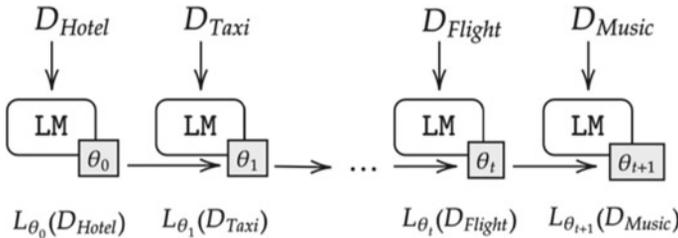


Fig. 6.9 Continual learning in a task-oriented dialogue system over multiple domains (Source Madotto et al. 2021)

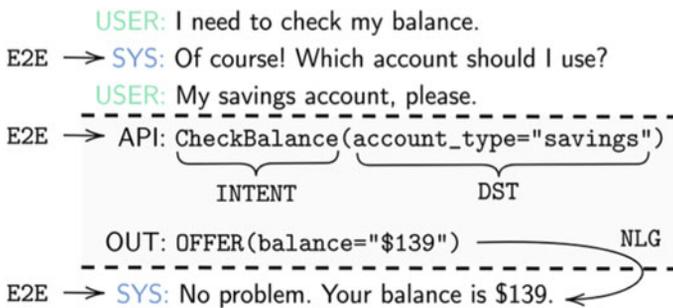


Fig. 6.10 Example of input-output pairs, for the four settings, INTENT, DST, NLG and end-to-end (E2E) (Source Madotto et al. 2021)

dialogue history except the last system utterance denoted as S . To distinguish between speakers, two special tokens are added at the beginning of every utterance i.e., `USER:` for the user utterance and `SYSTEM:` for the system utterance. Then, an API-call, denoted by S_{API} , is defined as the concatenation of the API-name (the user-intent) and its arguments (slot-value pairs) from the DST, as follows:

$$X_{API} = \underbrace{\mathbf{I}}_{\text{Intent}} (\underbrace{s_1 = v_1, \dots, s_k = v_p}_{\text{Slot-value pairs}}) \quad (6.19)$$

where \mathbf{I} is an intent (API-name), s_i is a slot-name and v_i is one of the possible values for slot s_i .

The return of the API-call is either an empty string or a speech-act, denoted as S_{OUT} (in the same format as the API-call). If S_{API} returns an empty string, it tells the model to use only the dialogue history to generate a response (discussed below). Similar to the dialogue history, two special tokens `API:` and `OUT:` are defined for triggering the model to generate an API-call and to distinguish the return of an API-call from the dialogue history respectively.

Given the aforementioned setting, the four tasks for CL are defined as follows:

1. The three modularized settings by their input-out pairs as:

$$\begin{aligned} H &\rightarrow \mathbf{I} && (\text{INTENT}) \\ H &\rightarrow \mathbf{I}(s_1 = v_1, \dots, s_k = v_p) && (\text{DST}) \\ \underbrace{\mathbf{I}(s_1 = v_1, \dots, s_k = v_p)}_{S_{OUT}} &\rightarrow S && (\text{NLG}) \end{aligned}$$

2. The end-to-end (E2E) setting

$$\begin{aligned} H &\rightarrow \underbrace{\mathbf{I}(s_1 = v_1, \dots, s_k = v_p)}_{S_{API}} \\ H + \underbrace{\mathbf{I}(s_1 = v_1, \dots, s_k = v_p)}_{S_{OUT}} &\rightarrow S \end{aligned}$$

Here, S_{OUT} can be empty in which case the model is assumed to map the dialogue history to the response directly ($H \rightarrow S$). An example of input-out pairs is shown in Fig. 6.10.

Based on the above setting, the dialogue dataset for a domain K denoted as $D_K = \{(X_i, Y_i)\}_i^n$ thus consists of general input-output pair (X_i, Y_i) from one of the four settings in consideration. To solve the task for each of the four settings, decoder-only Language Model (e.g., GPT-2) is trained following (Peng et al. 2020; Hosseini-Asl et al. 2020).

In the CL setting for ToDS as in (Madotto et al. 2021), *the task-id is provided during training, but not during testing*, i.e., during training the model is aware of which domain is

currently learning, but during testing, the model is evaluated *without* specifying the dialogue domain. Madotto et al. (2021) have compared various existing *regularization*, *replay* and *architectural* (also known as *parameter isolation*) based CL approaches (see Chap. 2 for more details) for solving the CL problem. Additionally, they have designed an architectural method based on residual adapters (Houlsby et al. 2019) and a replay-based strategy, called AdapterCL which outperforms most these baselines for the above four ToDS tasks, as discussed below.

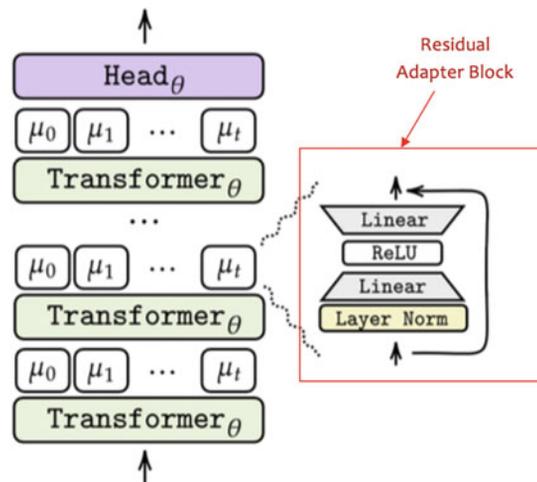
The AdapterCL model. The AdapterCL model parameterizes each task using Residual Adapters (Houlsby et al. 2019) and uses an entropy-based classifier to select which adapter to use at testing time. This method is designed for large pre-trained language models, e.g., GPT-2 where only the task-specific parameters in the adapters are trained and the original weights of the language model are left frozen. Figure 6.11 shows the high-level idea of the AdapterCL model.

Residual adapter block. Residual adapters are trainable parameters added on top of each transformer layer, which steers the output distribution of a pre-trained model without modifying its original weights. An adapter block consists of a Layer Normalization (Ba et al. 2016), followed by two linear layers with a residual connection. Given the hidden representation at layer l denoted as $H \in \mathbb{R}^{p \times d}$ of a transformer (Vaswani et al. 2017), where d is the hidden size and p is the sequence length, the residual adapter computes:

$$\text{Adapter}_{\mu_i}(H) = \text{ReLU}(\text{LN}(x)W_l^E)W_l^D + H, \quad (6.20)$$

where W_l^E and W_l^D are the trainable parameters of dimensions $d \times b$ and $b \times d$ respectively, $\text{LN}(\cdot)$ denotes the layer normalization and x is the input to the adapter block. The bottleneck dimension b is a tunable hyper-parameter that adjusts the capacity of the adapter according

Fig. 6.11 AdapterCL Model architecture (Source Madotto et al. 2021)



to the complexity of the target task. The set $\mu_i = \{W_0^E, W_0^D, \dots, W_L^E, W_L^D\}$ is defined as the set of parameters for the Adapter_{*i*} for a model with L layers.

To continuously or incrementally learn a new task with dataset D_t , a new Adapter is spawned with parameter μ_t , and then its parameters are trained by minimizing the negative log-likelihood over D_t as follows:

$$\mathcal{L}_{\mu_t}(D_t) = - \sum_j \sum_{i=0}^{|D_t|_{n+m}} \log p_{\mu_t}(x_i | x_0, \dots, x_{i-1}) \quad (6.21)$$

Note, the loss is optimized over μ_t to guarantee that each task is independently learned.

Perplexity-based classifier. In the aforementioned CL setting, the *task-id* is provided during training, thus each μ_t is optimized over D_t . But during testing, the *task-id* is *not provided*, thus the model has to predict which adapter to use to perform the task. Following (Wortsman et al. 2020), Madotto et al. (2021) have utilized the perplexity of each adapter over the input X as a measure of uncertainty. Thus, by selecting the adapter with the lowest perplexity, they select the most confident model to generate the output sequence.

Let α_t represent the confidence of the adapter t for the input X . Then, given the set of adapters parameterized by μ_0, \dots, μ_N , each of which trained respectively with D_0, \dots, D_N , and input sample X , α_t is computed as:

$$\alpha_t = \text{PPL}_{\mu_t}(X) \quad \forall t \in 1, \dots, N, \quad (6.22)$$

where, the perplexity of input sequence $X = \{x_0, \dots, x_n\}$ is defined as:

$$\text{PPL}_{\theta}(X) = \sqrt[n]{\prod_{i=1}^n \frac{1}{p_{\theta}(x_i | x_0, \dots, x_{i-1})}} \quad (6.23)$$

The computed α_i 's for all adapters $i \in \{0, \dots, N\}$ are then utilized to select the task-id t^* as:

$$t^* = \text{argmin } \alpha_0, \dots, \alpha_N \quad (6.24)$$

Note, the perplexity-based selector requires a linear number of forward passes with respect to the number of Adapters (Eq. 6.22), but it has the advantage of not requiring a further classifier (which can suffer from catastrophic forgetting and may require an episodic memory).

As a final note, it is useful to point out that in most existing works of CL, the models for all tasks share the same network so that they can share knowledge and transfer knowledge across tasks. The approach taken in (Madotto et al. 2021) trains a separate adapter module for each task, which clearly has no catastrophic forgetting, but also has no knowledge transfer. Its network size also grows with the number of tasks. There are several CL models (Ke et al. 2021a; Ke et al. 2021b) that enable all tasks to be learned in a single adapter module, which enables knowledge transfer and also have mechanisms to deal with catastrophic forgetting.

6.5.2 SL-AGENT

Recently, Zhang et al. (2022) have built SL-AGENT, a self-learning framework for building end-to-end ToDS in a more realistic changing environment setting with minimum or zero human annotations. It consists of a neural dialogue model and a pre-trained reward model, where the dialogue model generates responses and the reward model judges the quality of agent responses. Specifically, a data augmentation strategy is devised to construct positive and negative examples based on the given dialogue training corpus to endow the reward model with the capability to judge the quality of responses for unlabeled human-bot dialog logs.

The bot (including the dialogue model and reward model) is first trained with the same available training data and then, deployed to converse with real users to collect human-bot dialogue logs. After that, the bot is refined with the unlabeled human-bot dialogue logs via reinforcement learning, where the response quality is judged by the reward model. In this way, the bot can automatically adapt to unseen user behaviors without extra human annotation.

6.6 Summary

In this Chapter, we have discussed several methods to enable continual learning in task-oriented dialogue systems. We have highlighted two major limitations of existing ToDS, i.e., inability to deal with unknown intents and the problem of scalability over a large number of domains. For unknown intents detection and learning, we have discussed two recent approaches: the first approach is based on the Large Margin Cosine Loss (LMCL) and the Local Outlier Factor (LOF), and the second one learns Adaptive Decision Boundary (ADB) for open intent classification. In regard to the scalability issue, we have introduced a progressive neural network based model called ProgModel for continual learning for slot filling. We have also covered a discussion on continual learning for dialogue state tracking (DST), where we have described a model called Knowledge Preservation Networks (KPN) to solve the domain-lifelong learning problem in DST. Another model called Continual Prompt Tuning (CPT) is also introduced that enables knowledge transfer across tasks in the context of continual learning for DST. Next, we have discussed two recent works, Adaptive Regularized Prioritized Exemplar Replay (ARPER) and a transformer calibration for continual learning for natural language generation. Another model called AdapterCL has also been discussed that attempts to solve all ToDS sub-tasks jointly in a continual manner over multiple domains. Lastly, a recently proposed self-learning agent for ToDS called SL-AGENT is briefly discussed.

References

- A. Bendale, T. Boulton, Towards open world recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1893–1902
- M.M. Breunig, H-P. Kriegel, R.T. Ng, J. Sander, Lof: identifying density-based local outliers, in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (2000), pp. 93–104
- B. Byrne, K. Krishnamoorthi, S. Ganesh, M. Kale, Tickettalk: toward human-level performance with end-to-end, transaction-based dialog systems, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2021), pp. 671–680
- C-W. Goo, G. Gao, Y-K. Hsu, C-L. Huo, T-C. Chen, K-W. Hsu, Y-N. Chen, Slot-gated modeling for joint slot filling and intent prediction, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (2018), pp. 753–757
- C-H. Lee, H. Cheng, M. Ostendorf, Dialogue state tracking with a language model using schema-driven prompting, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), pp. 4937–4949
- Y. Deng, K. Chen, Y. Shen, H. Jin, Adversarial multi-label prediction for spoken and visual signal tagging, in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2019), pp. 3252–3256
- J. Devlin, M-W. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (2019), pp. 4171–4186
- G. Fei, B. Liu, Breaking the closed world assumption in text classification, in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2016), pp. 506–514
- Y. Guo, B. Liu, D. Zhao, Online continual learning through mutual information maximization, in *International Conference on Machine Learning* (PMLR, 2022b), pp. 8109–8126
- D. Guo, G. Tur, W-T. Yih, 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
- E. Haihong, P. Niu, Z. Chen, M. Song, A novel bi-directional interrelated model for joint intent detection and slot filling, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 5467–5471
- D. Hakkani-Tür, G. Tür, A. Celikyilmaz, Y-N. Chen, J. Gao, L. Deng, Y-Y. Wang, Multi-domain joint semantic frame parsing using bi-directional rnn-lstm, in *Interspeech* (2016), pp. 715–719
- M. Henderson, B. Thomson, S. Young, Word-based dialog state tracking with recurrent neural networks, in *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)* (2014), pp. 292–299
- E. Hosseini-Asl, B. McCann, W. Chien-Sheng, S. Yavuz, R. Socher, A simple language model for task-oriented dialogue. *Adv. Neural Inf. Process. Syst.* **33**, 20179–20191 (2020)
- N. Hounsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, S. Gelly, Parameter-efficient transfer learning for nlp, in *International Conference on Machine Learning* (PMLR, 2019), pp. 2790–2799
- D. Jurafsky, J.H. Martin, *Speech and language processing: an introduction to natural language processing*. *Comput. Linguist. Speech Recognit.* (2020)

- Z. Ke, B. Liu, H. Xu, N. Ma, L. Shu, Achieving forgetting prevention and knowledge transfer in continual learning. *Adv. Neural Inf. Process. Syst.* **34**, 22443–22456 (2021)
- Z. Ke, B. Liu, H. Xu, L. Shu, Classic: Continual and contrastive learning of aspect sentiment classification tasks, in *Proceedings of 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP-2021)* (2021b)
- S. Kim, S. Yang, G. Kim, S-W. Lee, Efficient dialogue state tracking by selectively overwriting memory in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020c), pp. 567–582
- J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, Others. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci.* **114**(13), 3521–3526 (2017)
- G. Kurata, B. Xiang, B. Zhou, M. Yu, Leveraging sentence-level information with encoder lstm for semantic slot filling, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (2016), pp. 2077–2083
- J. Lei Ba, J. Ryan Kiros, G.E. Hinton, Layer normalization (2016). [arXiv:1607.06450](https://arxiv.org/abs/1607.06450)
- W. Lei, X. Jin, M-Y. Kan, Z. Ren, X. He, 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)* (2018), pp. 1437–1447
- B. Lester, R. Al-Rfou, N. Constant, The power of scale for parameter-efficient prompt tuning, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), pp. 3045–3059
- Z. Lin, B. Liu, S. Moon, P.A. Crook, Z. Zhou, Z. Wang, Z. Yu, A. Madotto, E. Cho, R. Subba, Leveraging slot descriptions for zero-shot cross-domain dialogue state tracking, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2021a), pp. 5640–5648
- Q. Liu, P. Cao, C. Liu, J. Chen, X. Cai, F. Yang, S. He, K. Liu, J. Zhao, Domain-lifelong learning for dialogue state tracking via knowledge preservation networks, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021a), pp. 2301–2311
- Bing Liu, Ian Lane, Attention-based recurrent neural network models for joint intent detection and slot filling. *Interspeech* **2016**, 685–689 (2016)
- D. Lopez-Paz, M. Ranzato, Gradient episodic memory for continual learning, in *Advances in Neural Information Processing Systems* (2017), pp. 6470–6479
- L-M. Zhan, H. Liang, B. Liu, L. Fan, X-M. Wu, A.Y.S. Lam, Out-of-scope intent detection with self-supervision and discriminative training, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2021), pp. 3521–3532
- A. Madotto, Z. Lin, Z. Zhou, S. Moon, P.A. Crook, B. Liu, Z. Yu, E. Cho, P. Fung, Z. Wang, Continual learning in task-oriented dialogue systems, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021), pp. 7452–7467
- G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, Y. Dong et al., Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Trans. Audio, Speech Lang. Process.* **23**(3), 530–539 (2014)
- Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, Boi Faltings, Continual learning for natural language generation in task-oriented dialog systems. *Find. Assoc. Comput. Linguist.: EMNLP* **2020**, 3461–3474 (2020)
- N. Mrkšić, D. Ó Séaghdha, T-H. Wen, B. Thomson, 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)* (2017), pp. 1777–1788

- B. Peng, C. Li, J. Li, S. Shayandeh, L. Liden, J. Gao, Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model (2020). [arXiv:2005.05298](https://arxiv.org/abs/2005.05298)
- B. Peng, K. Yao, L. Jing, K-F. Wong, Recurrent neural networks with external memory for spoken language understanding, in *Natural Language Processing and Chinese Computing*. (Springer, 2015), pp. 25–35
- Avik Ray, Yilin Shen, Hongxia Jin, Iterative delexicalization for improved spoken language understanding. *Proc. Interspeech* **2019**, 1183–1187 (2019)
- A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks (2016). [arXiv:1606.04671](https://arxiv.org/abs/1606.04671)
- W.J. Scheirer, A. de Rezende Rocha, A. Sapkota, T.E. Boulton, Toward open set recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(7), 1757–1772 (2013)
- Y. Shan, Z. Li, J. Zhang, F. Meng, Y. Feng, C. Niu, J. Zhou, A contextual hierarchical attention network with adaptive objective for dialogue state tracking, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 6322–6333
- Y. Shen, X. Zeng, H. Jin, A progressive model to enable continual learning for semantic slot filling, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019), pp. 1279–1284
- L. Shu, H. Xu, B. Liu, Doc: deep open classification of text documents, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (2017), pp. 2911–2916
- S-A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert icarl: Incremental classifier and representation learning, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2017), pp. 2001–2010
- T-E. Lin, X. Hua, A post-processing method for detecting unknown intent of dialogue system via pre-trained deep neural network classifier. *Knowl.-Based Syst.* **186**, 104979 (2019)
- T-E. Lin, X. Hua, H. Zhang, Discovering new intents via constrained deep adaptive clustering with cluster refinement. *Proc. AAAI Conf. Artif. Intell.* **34**, 8360–8367 (2020)
- T-E. Lin, H. Xu, Deep unknown intent detection with margin loss, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019a), pp. 5491–5496
- A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017)
- Y. Wang, Y. Shen, H. Jin, A bi-model based rnn semantic frame parsing model for intent detection and slot filling, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (2018b), pp. 309–314
- H. Wang, Y. Wang, Z. Zhou, X. Ji, D Gong, J. Zhou, Z. Li, W. Liu, Cosface: large margin cosine loss for deep face recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018a), pp. 5265–5274
- J.D. Williams, M. Henderson, A. Raux, B. Thomson, A. Black, D. Ramachandran, The dialog state tracking challenge series. *AI Mag.* **35**(4), 121–124 (2014)
- Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, Ali Farhadi, Supermasks in superposition. *Adv. Neural Inf. Process. Sys.* **33**, 15173–15184 (2020)
- G. Yan, L. Fan, Q. Li, H. Liu, X. Zhang, X-M. Wu, A.Y.S. Lam, Unknown intent detection using gaussian mixture model with an application to zero-shot intent classification, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 1050–1060
- P. Yang, D. Li, P. Li, Continual learning for natural language generations with transformer calibration, in *Proceedings of the 26th Conference on Computational Natural Language Learning (CoNLL)* (2022), pp. 40–49

- K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, 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
- H. Zhang, X. Hua, T-E. Lin, Deep open intent classification with adaptive decision boundary. *Proc. AAAI Conf. Artif. Intell.* **35**, 14374–14382 (2021a)
- H. Zhang, X. Hua, T-E. Lin, R. Lyu, Discovering new intents with deep aligned clustering. *Proc. AAAI Conf. Artif. Intell.* **35**, 14365–14373 (2021b)
- C. Zhang, Y. Li, N. Du, W. Fan, S. Yu Philip, Joint slot filling and intent detection via capsule neural networks, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 5259–5267
- X. Zhang, B. Peng, J. Gao, H. Meng, Toward self-learning end-to-end task-oriented dialog systems, in *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2022), pp. 516–530
- X. Zhang, H. Wang, A joint model of intent determination and slot filling for spoken language understanding, in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), pp. 2993–2999
- Y. Zhao, Y. Zheng, Z. Tian, C. Gao, J. Sun, N.L. Zhang, Prompt conditioned vae: enhancing generative replay for lifelong learning in task-oriented dialogue, in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing* (2022), pp. 11153–11169
- Q. Zhu, B. Li, F. Mi, X. Zhu, M. Huang, Continual prompt tuning for dialog state tracking, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2022), pp. 1124–1137



In previous chapters, we have discussed how a chatbot can learn world knowledge (e.g., entities, facts, concepts) to generate more relevant responses and answer user questions (in Chap. 3), how it can improve its quality of response and avoid going out of context (in Chap. 5) and how it can acquire knowledge during conversation to understand user utterance better and serve user better than before (in Chaps. 4 and 6). All these qualities are essential to building a successful chatbot system that can respond satisfyingly and perform tasks well on users' behalf. However, another important aspect that it should possess is the qualities of *sensitivity*, *self-awareness* and *understanding of users' (interlocutors) characteristics* in order to best model its responses. These qualities are what separate us humans from machines. Specifically, the chatbot needs to learn users' behaviors, preferences, emotions, moods, opinions and situations and leverage these pieces of knowledge while crafting its responses. These conversational skills are essential to carry out *meaningful* and *engaging* conversations with users and make the chatbot more *human-like*.

There have been a large number of works on learning these conversational skills and proposing various effective solutions over the years. However, all these methods involve training models in *offline mode* using *pre-collected supervised* datasets and then, deploying them to work in practice. But, the issue is that the characteristics of the users whom these systems engage in conversations with can change over time. Users (as humans) are bound to go through various experiences in their daily lives which change their opinions, thoughts, beliefs, perspectives and thereby, their persona, likes and dislikes, emotions, moods, situations may also alter on timely basis. Thus, it becomes necessary for the chatbots to learn these characteristics (of users) continually during conversation and update its conversational skills over time.

In this chapter, we focus on how chatbots can learn these conversational skills continually during conversation. Although there exists myriads of related works on conversational

skills, to the best of our knowledge, we are not aware of any work addressing these tasks in the continual learning setting. We thus provide a brief survey of existing works in this chapter and discuss about possible extensions and future scopes of continual learning in this domain. In the following sections, we discuss learning user behaviour and preferences (in Sect. 7.1), learning emotions, moods and opinions (in Sect. 7.2) and situation-aware conversation modeling (in Sect. 7.3) and finally, conclude with a chapter summary (in Sect. 7.4).

7.1 Learning User Behaviors and Preferences

Learning behavioral traits and preferences of users and leveraging the knowledge in conversation modeling is an active area of research. Given a conversation context, the chatbot can learn the user’s behavioral and preference profile, i.e., whether a user feels more excited or gets annoyed while conversing on a particular topic, what his/her likes and dislikes are, etc. A number of works (Dong et al. 2022; Li et al. 2016b; Joshi et al. 2017; Zhang et al. 2018a; Yang et al. 2018b; Song et al. 2019; Luo et al. 2019; Liu et al. 2020b; Lin et al. 2021b) have been done focusing on building personalized dialogue systems. Personalized dialogue systems aim to learn and ground the user’s profile information to generate responses so that the user feels more engaged and connected with the chatbot and the conversation. Figure 7.1 shows an example adapted from Luo et al. (2019), where the chatbot recommends different restaurants for the same query issued by two different users based on their different profile information. In dialogue 1 (left side), the chatbot recommends “*The Fancy Pub*” which is a

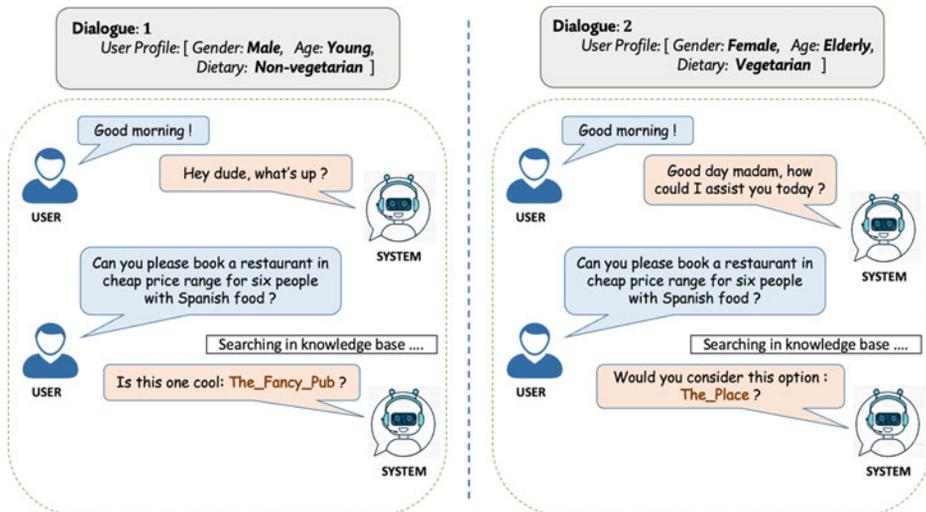


Fig. 7.1 Example dialogues showing personalized response generation for two user profiles (Source example adapted from Luo et al. 2019)

non-vegetarian pub mostly visited by *young* people. Whereas, in dialogue 2 (right side), the chatbot recommends “*The Place*”, a popular vegetarian option for elderly people to go for.

Among some of the noted contributions on personalized dialogue modelling, Li et al. (2016b) presented some persona-based models for handling the issue of speaker consistency in neural response generation. The proposed solution consists of a speaker model that encodes personas in distributed embeddings to capture individual characteristics such as background information and speaking style and a dyadic speaker-addressee model that captures properties of interactions between two interlocutors. Joshi et al. (2017) collected a new dataset of goal-oriented dialogues which are influenced by speaker profiles attached to them and analyzed the proposed modifications to an existing Memory Network based end-to-end dialogue system to enable personalization. Zhang et al. (2018a) have presented the task of making chit-chat dialogues more engaging by conditioning on the user profile information. They have collected a dataset called PERSONA-CHAT and used it to train models to condition on the given user profile information and information about the person to whom they are talking to generate personalized responses. Luo et al. (2019) proposed a personalized end-to-end model, namely PERSONALIZED MEMN2N to leverage personalization in goal-oriented dialogues. Song et al. (2019) have proposed a memory-augmented architecture to exploit the persona information from context and incorporated a conditional variational autoencoder model together to generate diverse and sustainable conversations. Liu et al. (2020b) have built P^2 BOT that incorporates mutual persona perception to enhance the quality of personalized dialogue generation. Lin et al. (2021b) have proposed a multi-lingual extension (in six different languages other than English) of PERSONA-CHAT (Zhang et al. 2018a), namely XPersona, to promote research on multi-lingual personalized dialogue systems. Qian et al. (2021b) have also proposed a large-scale dataset, Pchatbot, for personalized dialogue systems.

Other related works involve studying the problem in multi-party setting (Kumar et al. 2021), improving contextual coherence (Lee et al. 2022) and persona consistency (Kim et al. 2020a) in personalized dialogue agents, fusing persona information in retrieval-based chatbots (Gu et al. 2021b, 2019; Li et al. 2021), learning models from limited personalized data (Song et al. 2021), fusing persona information with emotions for response generation (Firdaus et al. 2020), designing personalized task-oriented dialogue systems (Zhang et al. 2020a), persona in persuasive dialogues (Wang et al. 2019; Yang et al. 2021b), gender biasness in personalized dialogue systems (Liu et al. 2020a). Besides, Hellou et al. (2021) have provided a review of technical methods for personalization in human-robot interactions, Polzehl et al. (2022) have leveraged persona for improving user experience in customer service applications and Yang and Flek (2021) have written a survey on user-centric text-to-text generation.

Aforementioned works primarily focus on how to leverage user’s persona information in modeling responses. There also have been some works (Wu et al. 2020; Gu et al. 2021a; Miehle et al. 2020; Qian et al. 2021a; Tiginova et al. 2019) focusing on learning user’s profile and preferences from conversation text. Tiginova et al. (2019) have proposed a number of

Hidden Attribute Models (HAMs) for inferring personal attributes, such as profession, age, family status from conversations. HAMs are trained on a per-predicate basis (e.g., for a subject’s profession or family status) to output rankings of object values for a given subject-predicate combination (e.g., ranking the doctor and nurse professions high when speakers talk about patients, emergency rooms, etc.).

Wu et al. (2020) have proposed a distant supervision based two-stage attribute extractor to automatically extract user attributes from dialogues. Given a user utterance, the work predicts user information that can be represented as a (*Subject, Predicate, Object*) triplet format. For example, (*I, live in, Florida*) is extracted from the user utterance “*I am originally from California but now I live in Florida for long*”. The problem of user attribute extraction is modeled as a pipeline of two tasks: *predicate prediction* task and *entity generation* task. The predicate prediction task first determines whether there is a predicate triggered by a user utterance—a multi-label classification problem. If there is a triggered predicate, then the entity generation task further generates the subject (the “who” information) and object (the “what” information) phrases to complete the whole user attribute.

Gu et al. (2021a) have studied the task of *Speaker Persona Detection* (SPD) for detecting speaker personas based on the plain conversational text. For this, the work has built a *Persona Match on PERSONA-CHAT* (PMPC) dataset and designed *utterance-to-profile* (U2P) matching network that searches out best-matched persona from a list of candidates given the conversational text. Qian et al. (2021a) have presented a retrieval-based personalized chatbot model, namely IMPChat, to learn an implicit user profile from the use’s dialogue history. The implicit user profile is learned through modeling user’s personalized language style and personalized preferences separately. Miehle et al. (2020) have proposed methods to estimate the communication style of spoken interaction, namely the stylistic variations *elaborateness* and *directness*, and investigated which type of input features to the estimator are necessary to achieve good performance.

All these approaches learn personalized conversation modeling in the offline setting. Often the persona information is provided with the labelled dataset and these approaches train models to generate personalized responses utilizing those persona information. However, in the post-deployment scenario, new users may appear with their own unique traits and preferences and persona information of existing users may change over time. Some efforts in this regard have been made in Madotto et al. (2019), which learns to quickly adapt to new personas by leveraging only a few dialogue samples collected from the same user. However, building chatbots that automatically detect new users, extract user profile dynamically from conversation context, store it in the knowledge base and leverage this information for response generation in a continual manner remain largely unexplored.

7.2 Learning Emotions, Moods and Opinions in Dialogues

Recognizing emotional state and sentiments of the user and leveraging it to generate empathetic responses can be useful in building therapeutic chatbots. A number of recent works (Li et al. 2020; Peng et al. 2019; Qiu et al. 2020; Sun et al. 2018; Zaranis et al. 2021; Zhou et al. 2018a) have studied the problem of emotional conversation modeling to realize this goal. Some of the notable works in this topic are as follows. Zhou et al. (2018a) have designed an encoder-decoder based model, namely *Emotional Chatting Machine* (ECM) to generate appropriate responses not only in content (relevant and grammatical) but also in emotion (emotionally consistent). Figure 7.2 shows an example of responses generated by ECM for various input emotions [shown in (.) next to “ECM” in Fig. 7.2] for a given user utterance along with the response generated by the basic Seq2Seq model (Vinyals and Le 2015) that does not use emotion information to generate its response.

Peng et al. (2019) have proposed a topic-enhanced emotional conversation generation model that incorporates both emotional factors and topic information into the conversation system. Colombo et al. (2019) have proposed an affect-driven dialog system, which generates emotional responses in a controlled manner using a continuous representation of emotions. The system has achieved this by modeling emotions at the word and sequence level using (1) a vector representation of the desired emotion, (2) an affect regularizer that penalizes neutral words, and (3) an affect sampling method that forces the neural network to generate diverse and emotionally relevant words.

Lin et al. (2019) have proposed an end-to-end framework, called *Mixture of Empathetic Listeners* (MoEL), for modeling empathy in dialogue systems. MoEL first captures the user emotions and outputs an emotion distribution. Based on this, it softly combines the output states of the appropriate Listener(s), which are each optimized to react to certain emotions, and generate an empathetic response. Rach et al. (2019) have presented an approach to model the opinion of a user towards a specific topic in a fine-grained way by using weighted bipolar argumentation graphs. Li et al. (2020) have proposed a multi-resolution adversarial model, EmpDG, that exploits both the coarse-grained dialogue-level and fine-grained token-level emotions, the latter of which helps to better capture the nuances of user emotions for generating more empathetic responses. They have also introduced an interactive adversarial learning framework which exploits the user feedback, to identify whether the generated responses evoke emotion perceptivity in dialogues. Majumder et al. (2020) have addressed the task of generating empathetic responses that mimic the emotion of the speaker while

Fig. 7.2 An example showing emotion-aware response generation (for various emotional states) in ECM (Courtesy: Zhou et al. 2018a)

User: Worst day ever. I arrived late because of the traffic.
Basic Seq2Seq: You were late.
ECM (<i>Like</i>): I am always here to support you.
ECM (<i>Happy</i>): Keep smiling! Things will get better.
ECM (<i>Sad</i>): It's depressing.
ECM (<i>Disgust</i>): Sometimes life just sucks.
ECM (<i>Angry</i>): The traffic is too bad!

accounting for their affective charge (positive or negative). They adopt the idea of MoEL to achieve the appropriate balance of emotions in positive and negative emotion groups. Additionally, they have also introduced stochasticity into the mixture at emotion group level for varied responses.

Shen and Feng (2020) have proposed a framework named *Curriculum Dual Learning* (CDL) which extends the emotion-controllable response generation to a dual task to generate emotional responses and emotional queries alternatively. CDL utilizes two rewards focusing on emotion and content to improve the duality. Additionally, it applies curriculum learning to gradually generate high-quality responses based on the difficulties of expressing various emotions. Zaranis et al. (2021) have introduced EmpBot, an end-to-end empathetic chatbot built based on a transformer pre-trained language model (T5) (Raffel et al. 2020).

Abro et al. (2022) have introduced a natural language understanding (NLU) framework for argumentative dialogue systems in the information-seeking and opinion building domain. Their proposed model is evaluated in an argumentative dialogue system that engages the user to inform him-/herself about a controversial topic by exploring pros and cons arguments and build his/her opinion towards the topic. Some recent works have also fused persona, opinions and emotions in response generation (Finch et al. 2020; Firdaus et al. 2020). Pamungkas (2019), Ma et al. (2020) have provided a comprehensive survey on emotional conversation modeling.

The aforementioned existing works on emotional conversation modeling have also trained their models in the offline setting. Similar to the idea of learning user behaviors and preferences in a continual learning setting, chatbots should also learn to recognize users' emotional states, moods and opinions and leverage the knowledge in a lifelong manner to improve its sensitiveness in carrying out dialogues with the user.

7.3 Modeling Situation-Aware Conversations

Understanding the situation and spatio-temporal context of a person and deciding the conversation strategy accordingly is key to having a good and effective human conversation. Continuously learning from the conversation history of the user provides a scope for the chatbot to learn user's conversation profile, e.g., what time of a day the user generally likes to talk or remains busy; understanding the spatial-temporal context of the user like whether the user is in a meeting or not, and so on can be useful in building *situation-aware proactive* chatbots and to improve user's overall conversation experience.

A large number of existing works have focused on building dialogue and interactive systems that are more *environment-aware* in which they are situated in. Among recent works on this topic, (Andrist et al. 2017; Bohus et al. 2021; Bohus and Horvitz 2009; Huang et al. 2021; Moon et al. 2020; Pejisa et al. 2014) have studied the problem of situated interaction systems design. Bohus and Horvitz (2009) have proposed computational models that allow spoken dialogue systems to handle multi-participant engagement in open, dynamic

environments, where multiple people may enter and leave conversations, and interact with the system and with others in a natural manner. Pejsa et al. (2014) have presented methods for estimating and communicating about different uncertainties (about properties of the world, people, and their intentions and actions) in situated interaction, leveraging the affordances of an embodied conversational agent.

Chai et al. (2014) have built a dialogue system that aims to mediate a shared perceptual basis during human-robot dialogue. Fang et al. (2015) have explored embodied collaborative models for referring expression generation to facilitate referential communication between humans and robots and mediate their differences in representing the shared environment. Liu and Chai (2015) have developed an optimization based approach that allows the robot to detect and adapt to perceptual differences in a shared environment.

Moon et al. (2020) have introduced *Situated Interactive MultiModal Conversations* (SIMMC) as a new direction aimed at training agents that take multimodal actions grounded in a co-evolving multimodal input context in addition to the dialog history. Huang et al. (2021) have proposed an end-to-end encoder-decoder model based on BART (Lewis et al. 2020) to solve the Dialogue State Tracking Challenge 9—*Situated Interactive MultiModal Conversations* (SIMMC) challenge, aimed at building virtual assistants that can handle multimodal inputs and perform multimodal actions. Kung et al. (2021) have leveraged multitask learning techniques to train a GPT-2 (Radford et al. 2019) based model on a more challenging dataset with multiple domains, multiple modalities, and more diversity in output formats. Bohus et al. (2021) have introduced *Platform for Situated Intelligence*, an open-source framework created to support the rapid development and study of multimodal, integrative-AI systems. Bohus and Horvitz (2019) have presented a comprehensive study on this topic.

Besides the work on situated interaction systems design, there are a few works on designing *socially-aware* chatbots (Zhao et al. 2016; Zhao 2019) in recent years. Motivated by the idea of modeling interpersonal rapport (which is an important function of human interaction and social intelligence), these works have focused on building dialogue systems that can build rapport with human in a task. Two types of scenarios have been studied: *cooperative* that includes personal assistants, peer tutoring etc. and *semi-cooperative* that includes negotiation (Lewis et al. 2017).

All these works mainly focus on unambiguously interpreting user's utterances or instructions considering the *environment as a context* and respond to the user accordingly. However, understanding the context of the user (human) as discussed before largely remain unexplored possibly due to the ethical and privacy issues/challenges (Henderson et al. 2018).

7.4 Summary

In this chapter, we have provided a motivation for real-world chatbots to have the ability to learn conversational skills continuously during conversation. In particular, we discuss about learning user behaviours and preferences, user emotions, moods and opinions, and

situations in dialogues to make the conversation more human-like and engaging to the user. As we have not found any specific work catering to these topics in the continual learning setting, we have mainly presented a brief survey of the recent advancements made in these topics and highlighted the scopes for performing continual learning in this domain as future work.

References

- W.A. Abro, A. Aicher, N. Rach, S. Ultes, W. Minker, G. Qi. Natural language understanding for argumentative dialogue systems in the opinion building domain. *Knowl-Based Syst.* **242**, 108318 (2022)
- W.A. Abro, A. Aicher, N. Rach, S. Ultes, W. Minker, G. Qi, Natural language understanding for argumentative dialogue systems in the opinion building domain. *Knowl-Based Syst.* **242**, 108318 (2022)
- S. Andrist, D. Bohus, E. Kamar, E. Horvitz, What went wrong and why? diagnosing situated interaction failures in the wild, in *International Conference on Social Robotics*. (Springer, 2017), pp. 293–303
- D. Bohus, S. Andrist, A. Feniello, N. Saw, M. Jalobeanu, P. Sweeney, A.L. Thompson, E. Horvitz, Platform for situated intelligence (2021). [arXiv:2103.15975](https://arxiv.org/abs/2103.15975)
- D. Bohus, E. Horvitz, Models for multiparty engagement in open-world dialog, in *Proceedings of the SIGDIAL 2009 Conference, The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2009), p. 10
- D. Bohus, E. Horvitz, Situated interaction, in *The Handbook of Multimodal-Multisensor Interfaces: Language Processing, Software, Commercialization, and Emerging Directions*, Vol. **3**, pp. 105–143 (2019)
- D. Bohus, E. Horvitz, Situated interaction. *The Handbook of Multimodal-Multisensor Interfaces: Language Processing, Software, Commercialization, and Emerging Directions* **3**, 105–143 (2019)
- J.Y. Chai, L. She, R. Fang, S. Ottarson, C. Littlely, C. Liu, K. Hanson, Collaborative effort towards common ground in situated human-robot dialogue, in *2014 9th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. (IEEE, 2014), pp. 33–40
- P. Colombo, W. Witon, A. Modi, J. Kennedy, M. Kapadia, Affect-driven dialog generation, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (2019), pp. 3734–3743
- W. Dong, S. Feng, D. Wang, Y. Zhang, I know you better: User profile aware personalized dialogue generation, in *International Conference on Advanced Data Mining and Applications*. (Springer, 2022), pp. 192–205
- R. Fang, M. Doering, J.Y. Chai, Embodied collaborative referring expression generation in situated human-robot interaction, in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction* (2015), pp. 271–278
- S.E. Finch, J.D. Finch, A. Ahmadvand, X. Dong, R. Qi, H. Sahijwani, S. Volokhin, Z. Wang, Z. Wang, J.D. Choi, et al., Emora: An inquisitive social chatbot who cares for you (2020). [arXiv:2009.04617](https://arxiv.org/abs/2009.04617)
- M. Firdaus, N. Thangavelu, A. Ekba, P. Bhattacharyya, Persona aware response generation with emotions, in *2020 International Joint Conference on Neural Networks (IJCNN)*. (IEEE, 2020), pp. 1–8

- J-C. Gu, Z-H. Ling, X. Zhu, Q. Liu, Dually interactive matching network for personalized response selection in retrieval-based chatbots, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019), pp. 1845–1854
- J-C. Gu, Z. Ling, Y. Wu, Q. Liu, Z. Chen, X. Zhu, Detecting speaker personas from conversational texts, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing* (2021a), pp. 1126–1136
- J-C. Gu, H. Liu, Z-H. Ling, Q. Liu, Z. Chen, X. Zhu, Partner matters! an empirical study on fusing personas for personalized response selection in retrieval-based chatbots, in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021b), pp. 565–574
- M. Hellou, N. Gasteiger, J.Y. Lim, M. Jang, H.S. Ahn, Personalization and localization in human-robot interaction: a review of technical methods. *Robotics* **10**(4), 120 (2021)
- M. Hellou, N. Gasteiger, J.Y. Lim, M. Jang, H.S. Ahn, Personalization and localization in human-robot interaction: a review of technical methods. *Robotics* **10**(4), 120 (2021)
- P. Henderson, K. Sinha, N. Angelard-Gontier, N.R. Ke, G. Fried, R. Lowe, J. Pineau, Ethical challenges in data-driven dialogue systems, in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society* (2018), pp. 123–129
- X. Huang, C.S. Tan, Y.B. Ng, W. Shi, K.H. Yeo, R. Jiang, J-j. Kim, Joint generation and bi-encoder for situated interactive multimodal conversations, in *AAAI 2021 DSTC9 Workshop* (2021)
- C.K. Joshi, F. Mi, B. Faltings, Personalization in goal-oriented dialog (2017). [arXiv:1706.07503](https://arxiv.org/abs/1706.07503)
- R. Kumar, D.S. Chauhan, G. Dias, A. Ekbal, Modelling personalized dialogue generation in multi-party settings, in *2021 International Joint Conference on Neural Networks (IJCNN)*. (IEEE, 2021), pp. 1–6
- J.Y. Lee, K.A. Lee, W.S. Gan, Improving contextual coherence in variational personalized and empathetic dialogue agents, in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2022), pp. 7052–7056
- M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer. Bart: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 7871–7880
- Q. Li, H. Chen, Z. Ren, P. Ren, Z. Tu, Z. Chen, Empdg: Multi-resolution interactive empathetic dialogue generation, in *Proceedings of the 28th International Conference on Computational Linguistics* (2020), pp. 4454–4466
- J. Li, M. Galley, C. Brockett, G. Spithourakis, J. Gao, W.B. Dolan, A persona-based neural conversation model, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2016b), pp. 994–1003
- J. Li, C. Liu, C. Tao, Z. Chan, D. Zhao, M. Zhang, R. Yan, Dialogue history matters! personalized response selection in multi-turn retrieval-based chatbots. *ACM Trans. Inf. Syst. (TOIS)* **39**(4), 1–25 (2021)
- J. Li, C. Liu, C. Tao, Z. Chan, D. Zhao, M. Zhang, R. Yan, Dialogue history matters! personalized response selection in multi-turn retrieval-based chatbots. *ACM Trans. Inf. Syst. (TOIS)* **39**(4), 1–25 (2021)
- Z. Lin, Z. Liu, G.I. Winata, S. Cahyawijaya, A. Madotto, Y. Bang, E. Ishii, P. Fung, Xpersona: evaluating multilingual personalized chatbot, in *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI* (2021b), pp. 102–112
- Z. Lin, A. Madotto, J. Shin, P. Xu, P. Fung. Moel: Mixture of empathetic listeners, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019), pp. 121–132

- C. Liu, J. Chai, Learning to mediate perceptual differences in situated human-robot dialogue, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29 (2015)
- Q. Liu, Y. Chen, B. Chen, J-G. Lou, Z. Chen, B. Zhou, D. Zhang, You impress me: Dialogue generation via mutual persona perception, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020b), pp. 1417–1427
- H.Liu, J. Dacon, W. Fan, H. Liu, Z. Liu, J. Tang, Does gender matter? towards fairness in dialogue systems, in *Proceedings of the 28th International Conference on Computational Linguistics* (2020a), pp. 4403–4416
- L. Luo, W. Huang, Q. Zeng, Z. Nie, X. Sun, Learning personalized end-to-end goal-oriented dialog. Proc. AAAI Conf. Artif. Intell. **33**, 6794–6801 (2019)
- Y. Ma, K.L. Nguyen, F.Z. Xing, E. Cambria, A survey on empathetic dialogue systems. Inf. Fusion **64**, 50–70 (2020)
- A. Madotto, Z. Lin, C-S. Wu, P. Fung, Personalizing dialogue agents via meta-learning, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 5454–5459
- N. Majumder, P. Hong, S. Peng, J. Lu, D. Ghosal, A. Gelbukh, R. Mihalcea, S. Poria, Mime: mimicking emotions for empathetic response generation, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2020), pp. 8968–8979
- J. Miehle, I. Feustel, J. Hornauer, W. Minker, S. Ultes, Estimating user communication styles for spoken dialogue systems, in *Proceedings of the Twelfth Language Resources and Evaluation Conference* (2020), pp. 540–548
- E.W. Pamungkas, Emotionally-aware chatbots: A survey (2019). [arXiv:1906.09774](https://arxiv.org/abs/1906.09774)
- T. Pejsa, D. Bohus, M.F. Cohen, C.W. Saw, J. Mahoney, E. Horvitz, Natural communication about uncertainties in situated interaction, in *Proceedings of the 16th International Conference on Multimodal Interaction* (2014), pp. 283–290, 2014
- T. Polzehl, Y. Cao, V. Iván Sánchez Carmona, X. Liu, C. Hu, N. Iskender, A. Beyer, S. Möller, Towards personalization by information savviness to improve user experience in customer service chatbot conversations, in *VISIGRAPP (2: HUCAPP)* (2022), pp. 36–47
- H. Qian, X. Li, H. Zhong, Y. Guo, Y. Ma, Y. Zhu, Z. Liu, Z. Dou, J-R. Wen, Pchatbot: a large-scale dataset for personalized chatbot, in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021b), pp. 2470–2477
- L. Qiu, Y. Shiu, P. Lin, R. Song, Y. Liu, D. Zhao, R. Yan, What if bots feel moods? in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020), pp. 1161–1170
- N. Rach, K. Weber, A. Aicher, F. Lingensfelder, E. André, W. Minker, Emotion recognition based preference modelling in argumentative dialogue systems, in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (IEEE, 2019), pp. 838–843
- A. Radford, W. Jeffrey, R. Child, D. Luan, D. Amodei, I. Sutskever et al., Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
- A. Radford, W. Jeffrey, R. Child, D. Luan, D. Amodei, I. Sutskever et al., Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
- C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 1–67 (2020)
- C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P.J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**, 1–67 (2020)

- L. Shen, Y. Feng, Cdl: curriculum dual learning for emotion-controllable response generation, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 556–566
- H. Song, Y. Wang, K. Zhang, W. Zhang, T. Liu, Bob: Bert over bert for training persona-based dialogue models from limited personalized data, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2021), pp. 167–177
- H. Song, W. Zhang, Y. Cui, D. Wang, T. Liu, Exploiting persona information for diverse generation of conversational responses, in *International Joint Conference on Artificial Intelligence* (2019)
- A. Tiginova, A. Yates, P. Mirza, G. Weikum, Listening between the lines: Learning personal attributes from conversations, in *The World Wide Web Conference* (2019), pp. 1818–1828
- O. Vinyals, Q. Le, A neural conversational model (2015). [arXiv:1506.05869](https://arxiv.org/abs/1506.05869)
- X. Wang, W. Shi, R. Kim, Y. Oh, S. Yang, J. Zhang, Z. Yu, Persuasion for good: Towards a personalized persuasive dialogue system for social good, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 5635–5649
- C-S. Wu, A. Madotto, Z. Lin, P. Xu, P. Fung, Getting to know you: User attribute extraction from dialogues, in *Proceedings of the 12th Language Resources and Evaluation Conference* (2020), pp. 581–589
- R. Yang, J. Chen, K. Narasimhan, Improving dialog systems for negotiation with personality modeling, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (2021b), pp. 681–693
- D. Yang, L. Flek, Towards user-centric text-to-text generation: a survey, in *Text, Speech, and Dialogue: 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6–9, 2021, Proceedings 24*. (Springer, 2021), pp. 3–22
- M. Yang, T. Wenting, Q. Qiang, Z. Zhao, X. Chen, J. Zhu, Personalized response generation by dual-learning based domain adaptation. *Neural Netw.* **103**, 72–82 (2018)
- E. Zaranis, G. Paraskevopoulos, A. Katsamanis, A. Potamianos. Empbot: A t5-based empathetic chatbot focusing on sentiments (2021). [arXiv:2111.00310](https://arxiv.org/abs/2111.00310)
- S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, J. Weston, Personalizing dialogue agents: I have a dog, do you have pets too? in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018a), pp. 2204–2213
- B. Zhang, X. Xiaofei, X. Li, Y. Ye, X. Chen, Z. Wang, A memory network based end-to-end personalized task-oriented dialogue generation. *Knowl.-Based Syst.* **207**, 106398 (2020)
- R. Zhao, *Socially-Aware Dialogue System*, Carnegie Mellon University, Ph.D. diss. (2019)
- R. Zhao, T. Sinha, A.W. Black, J. Cassell, Automatic recognition of conversational strategies in the service of a socially-aware dialog system, in *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2016), pages 381–392
- R. Zhao, *Socially-Aware Dialogue System* (Carnegie Mellon University, Ph.D. diss, 2019)
- H. Zhou, M. Huang, T. Zhang, X. Zhu, B. Liu, Emotional chatting machine: Emotional conversation generation with internal and external memory, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32 (2018a)



Conclusion and Future Directions

8

Dialogue systems are ubiquitous and being used to support a wide range of applications in businesses, healthcare, government, education, and entertainment. For example, most large companies have dialogue systems or chatbots for customer or technical support, guided selling, intranet employee question-answering, and healthcare consulting, etc. They also help students, teachers and scientists in all kinds of ways. Many such systems have been embedded in smart phones, cars, home appliances, service robots, etc. The applications will only get more widespread in the future. In fact, every machine learning system should also have a dialogue module so that the learning system can proactively interact with humans to learn more knowledge on the fly in a self-initiated manner. The second author had a personal experience in this regard when he consulted for a company that worked on self-driving cars. Once the team took a self-driving car for a test run. The car suddenly stopped and refused to move. At that time, every direction was clear. The team had no idea why and a human driver had to take over manually and drove the car back to the lab. After debugging, it was found that there was a pebble on the road detected by one of the sensors. If the self-driving car had been equipped with a dialogue system to help the car agent learn, it would have been very useful in this case. For example, the car can communicate with the engineers by saying something like *“I have detected an obstacle in front that I am not familiar with. What should I do?”* The engineers can just reply by saying *“this obstacle is very small and does not cause a problem. Please drive ahead.”* The system then can learn based on this instruction so that it will have no problem next time when it encounters a similar situation. In fact, dialogue systems are necessary for self-driving cars to communicate with passengers too for all kinds of purposes, e.g., to provide navigation instructions and to tell the car where to stop to drop off passengers or where to park.

Currently, user experiences have not been very satisfactory with most commercial dialogue systems and numerous improvements still need to be made. In this book, we have focused on the key aspect of *continuous learning* and *on-the-job learning after model deployment*, which is a new and emerging field. So far, limited research has been done. We are not aware of any existing deployed commercial dialogue systems that have the lifelong learning capability. We believe that one of the main reasons is that there are many technical challenges that are difficult to overcome. However, we believe that lifelong learning dialogue systems are necessary for the next generation dialogue systems. A system is not intelligent without the ability to learn many different types of knowledge, accumulate the knowledge over time, and use the knowledge to learn more and to learn better.

We also believe that it is time to put a significant amount of effort in the research of lifelong learning dialogue systems for several reasons: First, we believe some existing methods and their improved versions are ready for practical applications, e.g., some of those described in Chaps. 3 and 4. Although there are still challenges in continual learning itself such as catastrophic forgetting, there are usually engineering approaches that can get around of it in specific applications (see the discussion below about challenges). Second, there is already a huge number of commercial dialogue systems in practical use and many of them already have a large user base. It should be easy to conduct evaluation and user study if some form of lifelong learning capability is added to some of the systems. Third, lifelong learning has been an active research area in machine learning. Collaboration with machine learning researchers is likely to produce valuable results. Fourth, due to the fact that a large number of dialogue systems have been built, there is a fierce competition among companies building such systems. Adding a lifelong learning capability will give the company a competitive advantage. As we have shown in the book, lifelong learning can be included to solve many problems of the current dialogue systems.

This book surveyed the existing ideas and techniques. We hope the book can help promote and encourage more researchers to study the related problems and practitioners to build commercial lifelong learning dialogue systems. Below, we would like to highlight some challenging problems and future research directions. We categorize the challenges based on their topics into two groups.¹

Continual Learning

We start with continual learning, which is a core technology in lifelong learning dialogue systems. Recall that in traditional continual learning, the tasks and their training datasets are given. Some of the challenges and future research directions related to lifelong dialogue systems are as follows:

¹ We have discussed the challenges and scopes of continually learning of conversational skills in Chap. 7 and will not repeat them here.

1. **Overcoming catastrophic forgetting (CF).** Although there are a large number of papers that have been published on the topic, CF is still a major challenging problem that is hard to solve. Without solving this problem, it is hard to use a continual learning algorithm because it causes accuracy deterioration, which is often not acceptable in practice. We believe that with powerful pre-trained language models, this problem will be significantly alleviated. Kim et al. (2022, 2023) have made significant progresses on this both theoretically and algorithmically. Additionally, in the natural language domain, different applications often have a great deal of similarities, which makes the knowledge transfer easier, and forgetting may not be a major issue in many applications (Ke et al. 2021a, b).
2. **Knowledge transfer.** Compared to CF that has been studied extensively in machine learning and natural language processing, knowledge transfer is still on the back burner and has only had limited research attention. The main reason is that without effectively dealing with CF, there is no way to achieve knowledge transfer. However, some recent work have shown that knowledge transfer is possible for similar tasks (Ke et al. 2020, 2021a). Without knowledge transfer, the value of lifelong learning dialogue systems will be significantly diminished because during conversation, it is very difficult to acquire a large amount of training data for learning, which makes knowledge transfer necessary for effective learning. This topic needs urgent research in the dialogue systems context but it is quite promising as knowledge transfer has been very successful in the continual learning of sentiment analysis tasks (Ke et al. 2020, 2021a, b).
3. **Few-shot continual learning.** As discussed above, it is unlikely for a system to acquire a large volume of training data via interaction with the user on the fly. Thus, an effective and accurate few-shot continual/incremental learning method is necessary. Although there are some existing research on the issue, it is far from ready for practical use.
4. **Correctness of learned knowledge:** How to know whether a piece of past knowledge is correct is crucial for knowledge transfer in lifelong/continual learning (LL). Since we want to transfer the knowledge learned in the past to help future learning, incorrect past knowledge is very harmful. In many cases, LL is a continual bootstrapping learning process. Errors can propagate from previous tasks to subsequent tasks and result in more and more errors. This problem must be solved or mitigated. We human beings can solve this problem effectively. Even if mistakes are made initially, we can correct them later if new evidences appear. We can also backtrack and fix the errors along with the wrong inferences made based on the errors. An LL system should be able to do the same. The NELL system (Mitchell et al. 2015) deals with the problem by ensuring that the same item is extracted from multiple sources, by multiple strategies or meeting some type constraints. However, the existing methods are still quite rudimentary. Their recalls are low and can still get wrong knowledge.
5. **Learning new task completion skills.** Task-oriented chatbots are equipped with a pre-programmed and finite set of task completion skills. Building solutions to enable end users to use natural language dialogues to program their own chatbots or to endow them

with new skills after deployment will lead to personalized virtual assistants. This is also applicable to businesses that do not have in-house capabilities of building dialogue systems and license such systems from some AI software companies. In such a case, it is highly desirable that the businesses can add new services or skills to existing dialogue systems by themselves rather than going back to the companies from which they license the dialogue systems to request for the addition.

Continual Learning During Conversation

Next we focus on some of the challenging issues in learning on the job or continual learning during conversation.

1. **Identifying a learning task.** In general, when a novelty is detected, a new task will be formulated for incremental learning. Novelty detection is often the first step in on-the-job learning. We do not include the general problem of novelty detection here as it has been studied extensively (Pang et al. 2021; Yang et al. 2021). We focus on how to reliably decide when the system cannot understand an user utterance and the specific parts that it has difficulty with (which is novelty characterization). These issues are highly challenging and little research has been done. Without a reliable decision, the system will not be able to formulate a good task to learn and may even learn wrong knowledge and annoy the users by asking them wrong or even nonsense questions.
2. **Acquiring ground-truth training data on the fly.** One key feature of on-the-job learning is the interaction with human users to obtain ground-truth training data, which often involves multi-turn dialogues. This activity involves dynamically formulating an interaction strategy or a plan, e.g., deciding what to ask and when to ask. Although some limited research has been done in some simple cases (Mazumder et al. 2020, 2018), a general approach is still far from us. Additionally, multi-turn dialogue is a major challenge in dialogue systems research too. We are unaware of any such system for SOLA except CML (Mazumder et al. 2020), but CML is only for simple command learning.
3. **Novelty characterization and adaptation.** Novelty characterization is critical because it defines the characteristics used to recognize world state in the presence of novelty and determine the best response strategy. In the dialogue systems context, it mainly involves what the system can and cannot understand. We have given several examples of characterization and adaptation in dialogue systems in Chap. 2. However, little in-depth research has been done on the topics in the academic community. These topics are extremely challenging as they require the system to have a large amount of prior knowledge and an application domain world model, and the ability to reason based on the knowledge, the current user utterance and dialogue history.
4. **Knowledge representation and reasoning.** Learning during conversation need multiple functions, but it is not known what knowledge representation and reasoning scheme

best suits all modules and facilitates their integration and interaction. Important questions to be answered include what forms of knowledge are important, how to represent them, and what kinds of reasoning capabilities are necessary. So far, little research has been done to address these questions. Knowledge in existing systems is mainly represented based on the direct needs of specific learning algorithms or applications. They still do not have a reasoning ability.

5. **Learning to respond and internal evaluation system.** In order to learn to cope with unexpected situations, the system needs to learn by trial and error in the real situation, which needs an internal evaluation system to give rewards to actions and states and to be aware of *safety constraints*. For example, the system needs to know whether a response is good or not or whether the user is happy or not with the response.
6. **Leveraging large language models (LLMs).** LLMs such as ChatGPT and GPT-4 have recently revolutionized natural language understanding and dialogue systems. However, LLMs also have some major limitations for dialogues. First, LLMs contain only some general knowledge of the world, which may not be appropriate or sufficient for a highly specific and local application. In one of our projects, we tried to build a dialogue system for a specific healthcare related application using ChatGPT. However, the results were not satisfactory.

Second, natural conversations require the dialogue system to be *proactive*, i.e., to be able to ask questions and introduce related issues that are needed to solve a problem or to perform a task. However, LLMs are not good at that partly because they do not really understand what they have learned or generated and partly because there are not so much real conversation data of diverse types on the Web for LLMs to learn from as real conversations (e.g., with doctors, healthcare workers, customer care agents, or ticketing agents) are often not recorded. Even if they are recorded, they are likely to be highly private and not released to the public domain.

How to adapt and exploit the general knowledge and other capabilities in LLMs for domain-specific applications while also continually learning local and application-specific knowledge for dialogue generation without the need for too much domain-specific training data is an interesting research question. Some initial work has been done in this direction although not specifically for dialogue systems (Ke et al. 2022).

7. **Learning everything and everywhere.** As indicated in Chap. 2, every module or subsystem in Fig. 2.3 needs to learn continually, i.e., everywhere needs learning. Similarly, everything can be learned. For example, from each user's dialogue history with a chatbot, the system can learn whether a user feels more excited or gets annoyed while conversing on a particular topic, what he/she likes and dislikes. The chatbot can then utilize this user profile in modeling future conversations to make them more engaging with the user.
8. **Dealing with wrong knowledge from users.** As we proposed to learn new knowledge from end users, one major challenge is how to deal with intentional or unintentional wrong knowledge from users. For example, while providing demonstration of an action

or in a dialogue session with the agent, the user may perform a wrong action for a given input command or provide an incorrect feedback to the dialogue system. Then the system may display unintended behaviour afterward, which might even lead to safety or ethically issues.

Since chatbots usually work in a multi-user environment, such issues may be addressed through *cross-verification*. After acquiring a piece of new knowledge (a new command pattern or ground truth action) in an interaction session, the agent can store these new examples in a unverified knowledge buffer. Next, while interacting with some other users in future sessions to accomplish a related task, it can ask these users to verify the accumulated and unverified knowledge. Once verified for K times (by K different random users), the knowledge can be considered as trustworthy and removed from the unverified buffer and used in learning or chatting. This sounds like a reasonable idea, but no implementation or field testing has been done. So it is hard to know whether this strategy will work in a field test or evaluation.

9. **Revision of Knowledge.** Although strategies can be designed to cross-verify any knowledge learned from users, some wrong knowledge will inevitably be learned and stored in the knowledge base. How to detect and revise the wrong knowledge is a challenge. This requires a reasoning system to detect contradictions in the knowledge base and a knowledge revision method to correct the wrong knowledge and also all the consequences inferred from it. Further, it is inevitable that the system may misinterpret, generalize or otherwise assemble incorrect knowledge. A system must have a mechanism to detect and revise the inaccurate knowledge on its own. Again, little work has been done in this area.
10. **Dealing with safety and ethical issues.** The ability to learn continuously during conversations also comes with the problem of abusive language learning from end users. Learning user's behavior, situation and emotional profile and using the knowledge in unintended ways can become a risk to user privacy hacking and biased conversation modeling. Thus, *constrained conversational modeling* is needed to prevent unintended sharing and abusive use of user information.

This list of challenging problems or research directions is by no means exhaustive. As an emerging topic, current methods and systems are still rudimentary. But the journey of a thousand miles begins with the first step. The research area is a wide open field. A significant amount of research is still needed in order to make breakthroughs. Yet practical applications of dialogue systems call for this type of advanced learning in order to fundamentally advance dialogue systems and their applications. We envisage that in the near future, commercial lifelong learning dialogue systems will start to appear, which will revolutionize conversational AI. They will start simple with limited capabilities and narrow application scopes, but with time, they will mature and become more and more powerful.

References

- Z. Ke, B. Liu, H.X. Nianzu Ma, L. Shu, Achieving forgetting prevention and knowledge transfer in continual learning. *Adv. Neural Inf. Process. Syst.* **34**, 22443–22456 (2021)
- Z. Ke, B. Liu, H. Xu, L. Shu, Classic: continual and contrastive learning of aspect sentiment classification tasks, in *Proceedings of The 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP-2021)* (2021b)
- Z. Ke, Y. Shao, H. Lin, H. Xu, L. Shu, B. Liu, Adapting a language model while preserving its general knowledge, in *Proceedings of The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP-2022)* (2022)
- Z. Ke, B. Liu, X. Huang, Continual learning of a mixed sequence of similar and dissimilar tasks. *Adv. Neural Inf. Process. Syst.* **33**, 18493–18504 (2020)
- G. Kim, C. Xiao, T. Konishi, Z. Ke, B. Liu, A theoretical study on solving continual learning. **35**, 5065–5079 (2022)
- G. Kim, C. Xiao, T. Konishi, B. Liu, Learnability and algorithm for continual learning, in *Proceedings of Fortieth International Conference on Machine Learning (ICML-2023)* (2023)
- S. Mazumder, B. Liu, S. Wang, S. Esmailpour, An application-independent approach to building task-oriented chatbots with interactive continual learning, in *NeurIPS-2020 Workshop on Human in the Loop Dialogue Systems* (2020)
- S. Mazumder, N. Ma, B. Liu, Towards a continuous knowledge learning engine for chatbots (2018). [arXiv:1802.06024](https://arxiv.org/abs/1802.06024)
- T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, Never-ending learning, in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-15)* (2015)
- G. Pang, C. Shen, L. Cao, A. Van Den Hengel, Deep learning for anomaly detection: a review. *ACM Comput. Surv. (CSUR)* **54**(2), 1–38 (2021)
- J. Yang, K. Zhou, Y. Li, Z. Liu, Generalized out-of-distribution detection: a survey (2021). [arXiv:2110.11334](https://arxiv.org/abs/2110.11334)