Thèse n° 7214

Lifelong Machine Learning with Data Efficiency and Knowledge Retention

Présentée le 5 mars 2021

Faculté informatique et communications
Laboratoire d’intelligence artificielle
Programme doctoral en informatique et communications

pour l’obtention du grade de Docteur ès Sciences

par

Fei MI

Acceptée sur proposition du jury

Prof. K. Aberer, président du jury
Prof. B. Faltings, directeur de thèse
Prof. M. Huang, rapporteur
Dr L. Chen, rapporteuse
Prof. M. Jaggi, rapporteur

École
polytechnique
fédérale
de Lausanne

2021

Have the courage to change what's changeable,
have the magnanimity to accept what's not,
and have the wisdom to tell the difference.
— Kaifu Lee

To my family and friends ...

Acknowledgements

First of all, I would like to express my sincere gratefulness to my thesis director Prof. Boi Faltings for providing insightful supervision, support, and scientific freedom to pursue my research interests. I would also like to thank Prof. Minlie Huang for his guidance and care during the four month visit and valuable discussions on a lot of subjects afterwards. I am thankful to Prof. karl Aberer, Prof. Martin Jaggi, and Dr. Li Chen for taking their time to participate in my thesis defence, their knowledgeable comments, and helpful suggestions.

To current and former LIA and HCI lab members for helping with many aspects along the journey: Igor, Aleksei, Panayiotis, Naman, Diego, Adam, Ljubomir, Yubo, Onur, Claudiu, Goran, Florent, Aris, Marija. To Sylvie Thomet and Defferrard Patricia for their tireless help with all the administrative services. I am also very appreciative to many other collaborators and students with whom I worked on exciting projects: Tao, Kaicheng, Lingjing, Mengjie, Liangwei, Wanhao, Fengyu, Joshi—and many others.

My entire experience of the past few years would not have been as enjoyable without many friends who shared with me a great number of unforgettable and delightful moments. Besides some friends mentioned above, my warm gratitude goes to Ka Wai, Shan, Xiaodong, Shichao, Zhaowen, Xiujiang, Shuxuan, Yu, Danyu, as well as the "Lausanne Chinese basketball team": Jian, Yue, Simon, Alex—and many others.

Most importantly, I want to thank my family, and especially, my parents Mantian Mi and Qin Cheng. I am enormously grateful for their everlasting love, encouragement, and support in all my endeavours.

Lausanne, February 11, 2021 F. M.

Abstract

Artificial intelligence (AI) and machine learning (ML) have become de facto tools in many real-life applications to offer a wide range of benefits for individuals and our society. A classic ML model is typically trained with a large-scale static dataset in an offline manner. Therefore, it can not quickly capture new knowledge in non-stationary environments, and it is difficult to maintain long-term memory for knowledge learned earlier. In practice, many ML systems often need to learn new knowledge (e.g., domains, tasks, distributions, etc.) as more data and experiences are collected, which is referred to as a lifelong ML paradigm in this thesis. We focus on two fundamental challenges to achieve lifelong learning. The first challenge is to quickly learn new knowledge with a small number of observations, and we refer to it as **data efficiency**. The second challenge is to prevent an ML system from forgetting the old knowledge it has previously learned, and we refer to this challenge as **knowledge retention**. These two capabilities are crucial for applying ML to most practical applications. In this thesis, we study three important applications with these two challenges, including recommendation systems, task-oriented dialog systems, and the image classification task.

First, we propose two approaches to improve data efficiency for task-oriented dialog systems. The first proposed approach is based on Meta-learning, aiming to learn a better model parameter initialization from training data. It can quickly reach a good parameter region of new domains or tasks with a small number of labeled data. The second proposal takes a semi-supervised self-training approach to iteratively train a better model using sufficient unlabeled data when only a limited number of labeled data are available. We empirically demonstrate that both approaches effectively improve data efficiency to learn new knowledge. The second self-training method even consistently improves state-of-the-art large-scale pre-trained models.

Second, we tackle the knowledge retention challenge to mitigate the detrimental catastrophic forgetting issue when neural networks learn new knowledge sequentially. We formulate and investigate the “continual learning” setting for task-oriented dialog systems and recommendation systems. Through extensive empirical evaluation and analysis, we demonstrate the importance of (1) exemplar replay: storing representative historical data and replaying them to the model while learning new knowledge; (2) dynamic regularization: applying a dynamic regularization term to put flexible constraints on not forgetting previously learned knowledge in each model update cycle.

Lastly, we conduct several initial attempts to achieve both data efficiency and knowledge retention in a unified framework. In the recommendation scenario, we propose two approaches using different non-parametric memory modules to retain long-term knowledge. More importantly, the two proposed non-parametric predictions computed on top of them help learn and memorize new knowledge in a data-efficient manner. Apart from the recommendation scenario, we propose a probabilistic evaluation protocol in the widely studied image classification domain. It is general and versatile to simulate a wide range of realistic lifelong learning scenarios that require both knowledge retention and data efficiency for studying different techniques. Through experiments, we also demonstrate the benefit of data augmentation using Mixup in various realistic lifelong learning scenarios.

Keywords: lifelong machine learning, continual learning, few-shot learning, data efficiency, knowledge retention, catastrophic forgetting, task-oriented dialog, session-based recommendation, image classification.

摘要

人工智能和机器学习已经在许多现实应用场景中已经成为不可或缺的工具, 并为我们个体和社会提供了广泛的便利。 经典的机器学习模型通常在大规模静态数据中进行离线的训练，因此无法较好的应用在动态环境中，并且不能保持对早期学到的知识的长期记忆能力。 由于数据会不断被采集，经验会不断地扩充，实际场景中的机器学习系统常常需要不断学习新知识，例如新领域，新任务，新数据分布等等。此类需要不断学习新知识的场景是本论文讨论的“终生机器学习”框架。 我们主要探究终生机器学习场景下的两个核心挑战。第一个挑战是从少量的观察数据中学习新知识，我们把它叫做**数据效率（data efficiency）**。第二个挑战是防止机器学习模型遗忘其之前已经学过的旧知识，我们把它叫做**知识保留（knowledge retention）**。 以上两个挑战对于机器学习在许多实际场景中的应用是至关重要的。本论文中，我们主要研究具有上述两个挑战的三个重要应用场景，包括推荐系统，任务导向型对话系统，图像分类任务。

首先，我们针对任务导向型对话系统提出了两个提升数据效率的方法。第一个方法基于“元学习”（Meta-learning），该方法从训练数据中学一个更好的模型参数的初始化，使得模型能在当新领域或者新任务仅有少量标注数据的情况下更容易地学到比较好的参数。 第二个方法利用半监督的“自学习”（self-training）框架。该方法迭代地利用未标注数据来不断训练一个更好的模型，以提升仅有少量标注样本时的数据效率。实验证明以上两种方法都有效的提升了学习新知识时的数据效率，而且第二个自学习框架还进一步提升了当前最先进的的预训练语言模型的结果。

其次，我们探索了当神经网络模型在连续的学习新知识时的“知识保留”这个挑战，目的是防止神经网络模型的“灾难性遗忘” (catastrophic forgetting) 问题，我们在任务导向型对话系统和推荐系统中制定和研究了“持续学习”的设定。通过大量的实验评估和分析，我们证明了两方面技巧的有效性：（1）样本回放：在学习新知识的同时，存储具有代表性历史数据，并将其和新的训练数据进行拼接后再对模型进行训练；（2）动态正则：增加额外的动态正则项，每个周期对模型优化施加灵活的约束条防止遗忘先前学到的知识。

最后，我们针对在统一框架中同时解决数据效率和知识保留两个挑战，进行了一些初探。在推荐系统的应用场景中，我们提出两种非参数化存储模块以助于长期的知识保留。更重要的是，分别在它们之上提出的两种非参数化预测模型有助于高效地利用数据，从而很快地学习和记住新知识。 除了推荐的场景以外，我们还在被广泛研究的图像分类领域中，提出了一个基于概率的评估协议。该评估协议可以模拟实际终身学习场景中各种需要知识保留和数据效率的情况，因此其是可被用作研究各种技术的通用且全面的评估协议。通过实验，我们还在各种实际终身学习场景中证明了使用Mixup进行数据增强所带来的效果提升。

关键词：终生机器学习，持续学习，小样本学习，数据效率，知识保留，灾难性遗忘，任务导向型对话，序列推荐，图像分类。

Contents

List of Figures

List of Tables

1 Introduction

In the new era of big data, enormous signs of progress on artificial intelligence (AI) and machine learning (ML) offer a wide range of benefits for individuals and our society. However, the classic ML paradigm predominantly relies on fixed datasets and stationary environments. A learning algorithm runs over a given fixed dataset in a batch mode to learn a model's parameters, which often makes an independent and identical distribution (IID) assumption that learns different data points in isolation. Afterward, the trained model is evaluated w.r.t. another fixed testing dataset.

As time is irreversible, our world is never stationary. Humans can quickly learn new tasks with a limited number of observations. A shortcoming of the classic ML paradigm is that it can not deal with non-stationary environments. The classic ML paradigm emphasizes the final learning and optimization result from a fixed dataset. In contrast, the sequential learning process and the ability to quickly learn with the least amount of data are overlooked, which is referred to as the **data efficiency** challenge in this thesis. In many dynamic environments, an intelligent ML system needs to frequently learn new knowledge (e.g., tasks, domains, distributions, etc.) in a data-efficient manner based on continuous data input.

Another shortcoming of the classic ML paradigm is that it has no explicit memory modules. Thus it cannot retain the previously learned knowledge for a long time. Effective **knowledge retention** plays an essential role in the human learning process to (1) prevent newly learned knowledge from interfering with knowledge learned earlier and (2) accumulate the knowledge learned in the past to help the future learning process. Without this knowledge retention ability, ML systems can hardly reach human-level intelligence.

To tackled the above two **data efficiency** and **knowledge retention** challenges, we take a lifelong learning (Chen and Liu, 2018) perspective, which is also referred to as continual learning (Ring, 1994; Zenke et al., 2017a) or incremental learning (He et al., 2011; Castro et al., 2018) in some literature. It mainly describes a continuous learning process with the ability to retain and accumulate knowledge and use them to facilitate future learning as humans often do. As lifelong learning describes a learning paradigm rather than a concrete methodology, the research about lifelong learning needs to be grounded to specific use cases. This thesis studies data efficiency and knowledge retention challenges in different real-life applications, including recommendation systems, task-oriented dialog (ToD) systems, and computer vision systems.

The two aforementioned challenges of lifelong learning are crucial for the above three domains. In recommendation systems, we often have new items and new user preferences that arrive sequentially in time. Therefore, a recommendation model needs to quickly capture new and trendy patterns while maintaining old and static patterns that are still valid. In practice, a scalable ToD system also needs to expand its knowledge to new domains and new expressions frequently. Due to the high annotation cost in ToD systems, training ToD models in a data-efficient manner in new domains is practically important. At the same time, a ToD system's capability on knowledge gained previously should still be retained. In visual recognition systems, lifelong learning is also indispensable. For example, a robot needs to recognize new objects in new environments quickly and continuously while maintaining its knowledge of old ones. Apart from these three domains, lifelong learning is also crucial for many other real-life applications. Indeed, it is crucial for most practical ML and AI systems with temporality or scalability concerns.

The importance of the above two challenges can also be justified from a neuroscience perspective. The complementary learning theory (McClelland et al., 1995; O'Reilly and Norman, 2002; Blakeman and Mareschal, 2020) suggests that our brain uses a "neocortical" and a "hippocampal" learning system to achieve complex and complementary behaviors. The "neocortical" system (i.e., "neocortex") is characterized by a slow learning rate and is responsible for learning generalities. The "hippocampal" system (i.e., "hippocampus") allows short-term adaptation and aims to rapidly learn new information before integrated with the "neocortical" system. Altogether, the 'neocortical' and 'hippocampal' learning system aims to deal with the above knowledge retention and data efficiency challenges, respectively.

In Part I, we propose two approaches to improve data efficiency for ToD systems. The evaluation of the data efficiency ability is conducted in few-shot learning scenarios for a dialog system to learn some new tasks or domains with a small number of labeled data. The first proposed approach is based on a Meta-learning paradigm (Finn et al., 2017), which aims to learn a better model parameter initialization that can quickly reach a good parameter region of a target domain with a small number of labeled data. The second proposal takes a semi-supervised self-training (Kohonen, 1984; Lee, 2013) paradigm to iteratively train a better model using unlabeled data during fine-tuning the model in a new task with limited labeled data.

In Part II, we tackle the knowledge retention challenge to combat the detrimental catastrophic forgetting issue when neural networks need to learn a sequence of tasks. Applications of task-oriented dialog systems and recommendation systems are considered accordingly. Through extensive analysis with different neural networks in different applications, we demonstrate the importance of *storing* representative data in the previous tasks and *replaying* them to the model while learning a new task. Furthermore, we also reveal the complementary benefits obtained by applying a dynamic regularization term to put stronger constraints on not forgetting previously learned knowledge.

In Part III, we propose several initial attempts to deal with data efficiency and knowledge retention challenges in a unified framework. Chapter 6 focus on the domain of recommendation systems. In the first proposed method (NECT), deep neural networks are not used. Instead, a variable-order Markov prediction model (Begleiter et al., 2004) is applied on top of a non-parametric memory component called context tree (Willems et al., 1995; Garcin et al., 2013). In the second proposed approach (MAN), a neural model's prediction is adjusted by taking into account the k-nearest neighbors (KNN) retrieved from a non-parametric key-value memory component similar to Grave et al. (2017b,a); Tu et al. (2018); Orhan (2018). The non-parametric memory component helps to retain old knowledge better, and the good *memorization* ability (Cohen et al., 2018; Khandelwal et al., 2020) of KNN helps to quickly capture new knowledge with limited observations to improve data efficiency. We empirically demonstrate that these two proposed methods achieve promising recommendation performance, especially on new and infrequent items. Chapter 7 proposes a probabilistic formulation to simulate a wide range of realistic lifelong learning scenarios for the image classification task. We also demonstrate that the data augmentation technique based on Mixup (Zhang et al., 2018) helps to train a more robust model for better knowledge retention and data efficiency.

In the remaining part of this chapter, we explain our motivations; overview several related learning paradigms; highlight our contributions, and outline the structure of this thesis.

1.1 Motivation

The initial motivation of this thesis is originated from the dynamic environments in recommendation scenarios. In many domains, such as e-commerce, social media, and news platforms, new items and user preferences frequently appear and override old ones. Therefore, a good recommendation model needs to quickly captures these new items or distribution shifts. This is referred to as the data efficiency challenge in this thesis. To solve this problem, we propose two approaches in Chapter 6 using a context tree structure and a k-Nearest Neighbor principle respectively to capture new preferences better while maintaining old ones.

After exploring several approaches in the recommendation domain, we found that a similar data efficiency challenge is also crucial in many other applications. Therefore, we extend our research boundary to a broader range of tasks. More specifically, we found that the labeling cost in the ToD system is very high, and existing models are mostly trained for a limited number of domains with sufficient annotations. Therefore, enabling a ToD system to learn domains with insufficient annotations becomes a practically important topic. With this objective in mind, we propose two approaches using Meta-learning and self-training in Chapter 2 and 3 respectively. Recently, Meta-learning is widely used in various few-shot learning tasks in the computer vision domain. We are pioneers in applying it to the natural language domain, especially in ToD systems. The second self-training proposal is mainly motivated by the fact that there are sufficient unlabeled dialog data in practice. To this end, we proposed a self-training pipeline to utilize the unlabeled data for gradually training a better model. We empirically demonstrate that it achieves complimentary benefits on top of state-of-the-art large-scale pre-trained models (Devlin et al., 2019a; Wu et al., 2020) which have been recently shown to be good few-shot learners (Brown et al., 2020).

In the process of developing solutions for the initial data efficiency challenge, the scope of our work has expanded, and motivation has further evolved. We found that knowledge retention is also an important issue in most practical applications. It is crucial to achieving lifelong learning as an ML system often needs to continually learn new knowledge (e.g., domains, tasks, or distributions). For example, when a ToD system needs to learn new domains with limited annotations, its capability on old domains still needs to be retained. A similar concern also exists in recommendation systems to retain old user preferences when they are still valid. Therefore, we study the knowledge retention challenge in these two application scenarios to mitigate the detrimental catastrophic forgetting issue in Part II (Chapter 4 & 5).

Based on our several explorations in different applications, we found that the lifelong learning setting and its corresponding issues are domain-specific and task-specific, and existing evaluation protocols have several limitations. Therefore, we propose a probabilistic evaluation protocol (GCCL) for the widely studied image classification task to simulate a wide range of realistic lifelong learning scenarios that require both knowledge retention and data efficiency in Chapter 7. GCCL aims to serve as a versatile evaluation protocol to studied different lifelong learning techniques. Through extensive experiment evaluation and analysis, we also demonstrate the benefit of data augmentation to improve both knowledge retention and data efficiency.

1.2 Related Learning Paradigms

As described before, we focus on knowledge retention and data efficiency challenges in lifelong learning. Several other learning paradigms have similar considerations, and we discuss some of their similarities and differences to ours at a high level below.

Transfer learning (Pan and Yang, 2010; Weiss et al., 2016) is a widely studied topic in ML, which describes a process of "knowledge transfer" from some source domains to a target domain with potentially limited labeled data to address the data efficiency challenge. The mainstream approaches identify or learn domain-invariant features or representations to help the performance of the target domain. Despite its great importance to the ML community, transfer learning does not consider a sequential learning process because the knowledge transfer is unidirectional and is a one-time process.

Multi-task learning (Ruder, 2017; Zhang and Yang, 2017) aims to learn multiple related tasks simultaneously to achieve better performance than learning them separately. The main idea is to utilize the relevant information shared by multiple tasks, and it also alleviates overfitting the individual task for better generalization ability. Like transfer learning, multi-task learning can be categorized as a classic learning paradigm without a sequential learning process. Multiple tasks can be seen as one bigger task, and they are learned all at once.

Online learning (Saad, 1998; Bottou and Cun, 2003) is a learning paradigm where the model is trained on data points that arrive in a sequential order. It is often used when it is computationally infeasible to train over the entire dataset, or some data only get available later in time. Retaining the knowledge that the model has previously seen and learning new knowledge in a data-efficient manner are still important topics for online learning. However, online learning often learns a model for a single task or a single distribution using the notion of "regret" from the optimal model so far. In this sense, it is still similar to the classic learning paradigm because it does not concern learning new knowledge from new tasks or new distributions.

1.3 Our Contributions

To address data efficiency and knowledge retention challenges outlined above, we propose several novel techniques for different applications and learning scenarios. Our main contributions in this thesis are the following:

- To improve the data efficiency for ToD systems, we propose two approaches in Chapter 2 and 3 respectively. In a nutshell, the first Meta-learning approach leverages *meta tasks* to learn a better parameter initialization from source domains such that it can learn new domains with high data efficiency. The second proposed approach utilizes the sufficient unlabeled data in ToD systems in a semi-supervised learning setup to overcome the difficulty of the limited number of labeled data. A new self-training pipeline with a novel text augmentation technique is proposed to further improve state-of-the-art pre-trained models for different few-shot learning tasks in ToD systems.

- To tackle the knowledge retention challenge in ToD systems, we study a continual learning setup in Chapter 4 to sequentially learn multiple domains for the natural language generation module. We identify the detrimental catastrophic forgetting issue when the model is evaluated on all seen domains (i.e., the past counts). As a solution, we propose a method called ARPER with adaptive regularization and periodically replaying a small number of representative samples from the past. We empirically demonstrate that ARPER effectively mitigates the catastrophic forgetting issue for different base models, and it performs comparably to using all historical data every time. Our proposals and promising results obtained for data efficiency and knowledge retention challenges may shed light on further directions towards building more scalable ToD systems.

- For recommendation systems, precisely the session-based recommendation task, we also formulate a continual learning setup to study the knowledge retention challenge. We find that the catastrophic forgetting issue is less severe because (1) some old preference patterns are still valid as previous items often reappear, and (2) the evaluation in recommendation systems is only w.r.t. future observations and obsolete patterns are ignored (i.e., only the future counts). The proposed method ADER in Chapter 5 using adaptive regularization and exemplar replay achieves promising results. It even performs better than training the model using all historical data every time.

- Another critical issue in recommendation systems is the data efficiency challenge, especially in very dynamic environments. To this end, we study how to quickly capture new preference patterns on new items without forgetting old ones that are still valid. Two methods (NECT and MAN) are proposed in Chapter 6, and the main idea is to use a non-parametric method for both memorization and prediction. The non-parametric nature of the two proposed methods helps them effectively memorize and predict new patterns that are still not statistically significant for a big neural model to capture.

- Apart from the investigation into ToD and recommendation systems, we also study the above two challenges for the widely studied image classification task in Chapter 7. We extend the commonly used class-continual learning setup (Rebuffi et al., 2017; Castro et al., 2018) with a probabilistic formulation to simulate a wide range of realistic scenarios. Through benchmarking various baselines, we identify the benefit of data augmentation using Mixup (Zhang et al., 2018) in different realistic class-incremental learning scenarios to address both knowledge retention and data efficiency challenges.

1.4 Organization of the Thesis

This thesis consists of three major parts. The first part (Chapter 2 & 3) is centred around dealing with the data efficiency challenging. Two approaches using Meta-learning and self-training are proposed in few-shot learning scenarios for ToD systems. The second part (Chapter 4, 5, & 7) focuses on dealing with the knowledge retention challenge to mitigate catastrophic forgetting when a neural model learns a sequence of tasks sequentially. Techniques and realistic learning scenarios are studied for ToD systems, recommendation systems. The third part (Chapter 6) attempts to put together both the data efficiency and knowledge retention challenges for recommendation systems and the image classification task.

Specifically, different chapters of this thesis are organized in the following way:

- In Chapter 2, we propose a Meta-learning method call Meta-NLG for the few-shot learning task of the natural language generation module in ToD systems. We demonstrate that Meta-NLG learns new language patterns in new domains with limited annotations very well.
- In Chapter 3, we propose a self-training approach to further improve the strong large-scale pre-trained models for four different downstream tasks in ToD. To gradually train a stronger *Student* model using unlabeled data, we demonstrate the benefit of (1) iteratively assigning confident pseudo-labels to unlabeled data and (2) proper text augmentation techniques to improve the model's generalization ability.
- In Chapter 4, we study how to mitigate the catastrophic forgetting challenge while learning a sequence of domains for the natural language generation module in ToD systems. A method called ARPER is proposed by storing representative samples with adaptive regularization.
- In Chapter 5, we formulate a continual learning setup for recommendation systems and study the knowledge retention issue. The proposed method (ADER) by storing representative samples and adaptive regularization similar to Chapter 4 also achieve promising performance.
- In Chapter 6, we deal with knowledge retention and data efficiency together in the recommendation domain. We demonstrate that the two proposed methods (NECT and MAN) achieve promising results by making use of non-parametric memorization and prediction.
- In Chapter 7, we propose a probabilistic formulation (GCCL) to simulate a wide range of realistic continual learning scenarios that require both knowledge retention and data efficiency. We notice that different datasets in different applications have different characteristics. Therefore, the proposed GCCL intends to serve as a

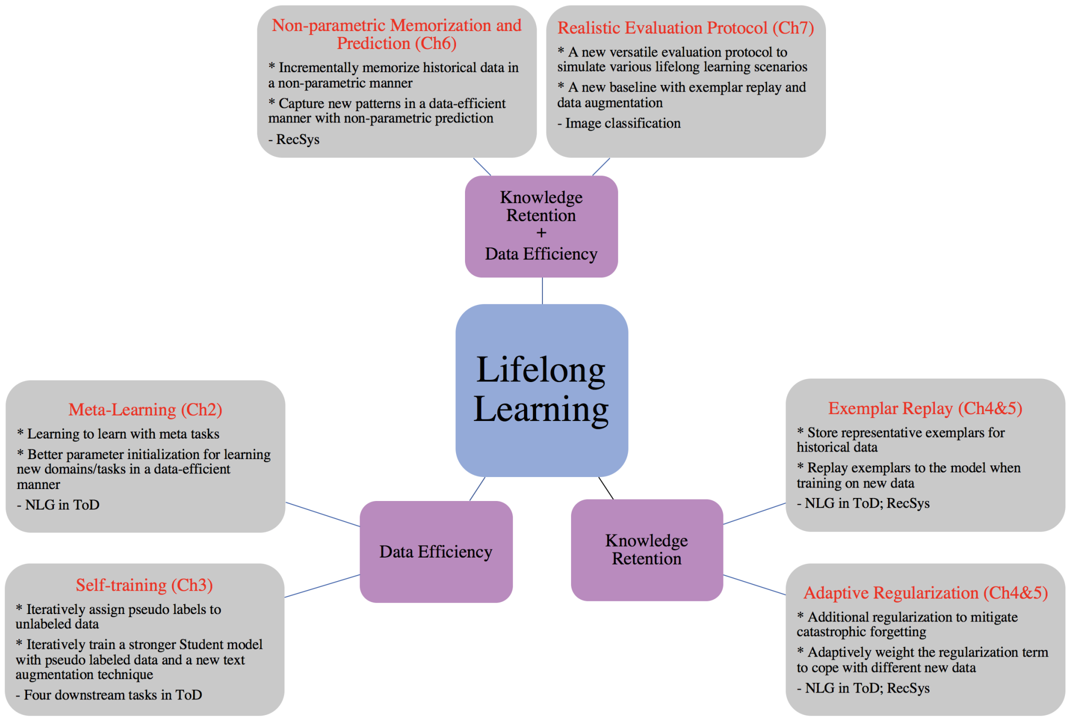

Non-parametric Memorization and Prediction (Ch6)
* Incrementally memorize historical data in a non-parametric manner
* Capture new patterns in a data-efficient manner with non-parametric prediction
- RecSys
Realistic Evaluation Protocol (Ch7)
* A new versatile evaluation protocol to simulate various lifelong learning scenarios
* A new baseline with exemplar replay and data augmentation
- Image classification
Knowledge Retention + Data Efficiency
Lifelong Learning
Meta-Learning (Ch2)
* Learning to learn with meta tasks
* Better parameter initialization for learning new domains/tasks in a data-efficient manner
- NLG in ToD
Data Efficiency
Knowledge Retention
Exemplar Replay (Ch4&5)
* Store representative exemplars for historical data
* Replay exemplars to the model when training on new data
- NLG in ToD; RecSys
Self-training (Ch3)
* Iteratively assign pseudo labels to unlabeled data
* Iteratively train a stronger Student model with pseudo labeled data and a new text augmentation technique
- Four downstream tasks in ToD
Adaptive Regularization (Ch4&5)
* Additional regularization to mitigate catastrophic forgetting
* Adaptively weight the regularization term to cope with different new data
- NLG in ToD; RecSys

Figure 1.1 – Overview of the topics studied in this thesis. Highlights of different chapters (gray) in three parts (purple) are illustrated, and the last item in each gray box indicate the application(s) we studied.

versatile evaluation protocol to studied different techniques. We also demonstrate the benefit of data augmentation for both challenges.

- Chapter 8 concludes the thesis with a summary and future research directions.

In Figure 1.1, we illustrate the topics in three parts (purple) to achieve lifelong learning in this thesis. Key contributions and the studied application(s) of different chapters are illustrated in gray boxes.

Data Efficiency Part I

2 Meta-Learning Improves Data Efficiency in ToD Systems

2.1 Introduction

As an essential part of a task-oriented dialogue system (Wen et al., 2017a), the task of natural language generation (NLG) is to produce a natural language utterance containing the desired information given a *semantic representation*. Conventional methods using hand-crafted rules often generate monotonic utterances, and it requires a substantial amount of human engineering work. Recently, various neural approaches (Wen et al., 2015b; Tran and Nguyen, 2017; Tseng et al., 2018) have been proposed to generate accurate, natural, and diverse utterances. However, these methods are typically developed for particular domains. Moreover, they are often data-intensive to train. The high annotation cost prevents developers from building their own NLG component from scratch. Therefore, it is beneficial to train an NLG model that can be generalized to other NLG domains or tasks with a reasonable number of annotated data. That is, the NLG model is supposed to learn new domains or tasks in a data-efficient manner. This is referred to as *low-resource* NLG task in this chapter.

Recently, some methods have been proposed for low-resource NLG tasks. Apart from the simple data augmentation trick (Wen et al., 2016), specialized model architectures, including conditional variational auto-encoders (CVAEs, (Tseng et al., 2018; Tran and Nguyen, 2018a,b)) and adversarial domain adaptation critics (Tran and Nguyen, 2018a), have been proposed to learn domain-invariant representations. Although promising results were reported, we found that datasets used by these methods are simple, which tend to enumerate many slots and values in an utterance without many linguistic variations. Consequently, over-fitting the slots and values in the low-resource target domain could even outperform those versions trained with rich source domain examples (Tran and Nguyen, 2018b). Fortunately, a new large-scale human-written dialog dataset (MultiWOZ, (Budzianowski et al., 2018)) contains a variety of domains and linguistic patterns, which

This chapter is based on the paper (Mi et al., 2019) published in the International Joint Conference on Artificial Intelligence (IJCAI, 2019).

allows us to conduct extensive and meaningful experimental analysis for low-resource NLG tasks. We are the first to ground the research on low-resource NLG task on this dataset.

In this chapter, we take a Meta-learning perspective to solve the low-resource NLG task. Meta-learning or learning-to-learn, which can date back to some early works (Naik and Mammone, 1992), has recently attracted extensive attention. A fundamental problem is "fast adaptation to new and limited observation data". In pursuing this problem, there are three categories of meta-learning methods:

- **Metric-based**: the idea is to learn a metric space and then use it to compare low-resource testing samples to rich training samples. The representative works in this category include Siamese Network (Koch, 2015), Matching Network (Vinyals et al., 2016), Memory-augmented Neural Network (Santoro et al., 2016a), Prototype Net (Snell et al., 2017), and Relation Network (Sung et al., 2018).
- **Model-based**: the idea is to use an additional meta-learner to learn to update the original learner with a few training examples. (Andrychowicz et al., 2016) developed a meta-learner based on LSTMs. Hypernetwork (Ha et al., 2017), MetaNet (Munkhdalai and Yu, 2017), and TCML (Mishra et al., 2017) also learn a separate set of representations for fast model adaptation. (Ravi and Larochelle, 2017) proposed an LSTM-based meta-learner to learn the optimization algorithm (gradients) used to train the original network.
- **Optimization-based**: the optimization algorithm itself can be designed in a way that favors fast adaption. Model-agnostic meta-learning (MAML, (Finn et al., 2017; Yoon et al., 2018; Gu et al., 2018)) achieved state-of-the-art performance by directly optimizing the gradient towards a good parameter initialization for easy fine-tuning on low-resource scenarios. It introduces no additional architectures nor parameters. Reptile (Nichol and Schulman, 2018) is similar to MAML with only first-order gradient. In this chapter, we propose a generalized meta optimization method based on MAML to directly solve the intrinsic learning issues of low-resource NLG tasks.

Metric-based or model-based techniques are mainly designed for classification tasks, while the optimization-based approach serves as a more generalized approach with a superior performance achieved by MAML. In this chapter, we propose a generalized meta optimization method based on MAML to directly solve the intrinsic learning issues of low-resource NLG tasks. Specifically, we proposed a generalized NLG algorithm called Meta-NLG based on MAML by viewing languages in different domains or dialog act intents as separate *Meta NLG tasks*. Following the essence of MAML, the goal of Meta-NLG is to learn a better initialization of model parameters that facilitates fast adaptation to new low-resource NLG scenarios from *Meta NLG tasks* rather than from individual data point. As Meta-NLG is model-agnostic, as long as the model can be optimized by gradient descent, we could apply it to any existing neural NLG models to optimize them in a way that adapts better and faster to new low-resource tasks. We extensively evaluate Meta-NLG on the largest human written multi-domain dataset (MultiWoz) with various low-resource NLG scenarios. Results show that Meta-NLG significantly outperforms other optimization methods in various configurations. We further analyze the superior performance of Meta-NLG, and show that it indeed adapts much faster and better.

2.2 Background on NLG for ToD

The NLG component of task-oriented dialog systems is to produce natural language utterances conditioned on a *semantic representation* called dialog act (DA). Specifically, the dialog act $\mathbf{d}$ is defined as the combination of *intent* $\mathbf{I}$ and a set of slot-value pairs $S(\mathbf{d}) = \{(s_i, v_i)\}_{i=1}^{p}$:

$$\mathbf{d} = [\underbrace{\mathbf{I}}_{\text{Intent}}, \underbrace{(s_1, v_1), \ldots, (s_p, v_p)}_{\text{Slot-value pairs}}], \tag{2.1}$$

where p is the number of slot-value pairs. Intent $\mathbf{I}$ controls the utterance functionality, while slot-value pairs contain information to express. For example, "*There is a restaurant called [La Margherita] that serves [Italian] food.*" is an utterance corresponding to a DA "*[Inform, (name=La Margherita, food=Italian)]*"

Conditioned on a DA, a neural NLG model generates an utterance containing the desired information word by word. For a DA $\mathbf{d}$ with the corresponding ground truth utterance $\mathbf{Y} = (y_1, y_1, ..., y_K)$, the probability of generating $\mathbf{Y}$ is factorized as below:

$$f_\theta(\mathbf{Y}, \mathbf{d}) = \prod_{k=1}^{K} p_{y_k} = \prod_{k=1}^{K} p(y_k | y_{<k}, \mathbf{d}, \theta), \tag{2.2}$$

where f_θ is the NLG model parameterized by θ, and p_{y_k} is the output probability (i.e. softmax of logits) of the ground truth token y_k at position k. The typical objective function for an utterance $\mathbf{Y}$ with DA $\mathbf{d}$ is the average cross-entropy loss w.r.t. all tokens in the utterance (Wen et al., 2015b,c; Tran and Nguyen, 2017; Peng et al., 2020b):

$$\mathcal{L}_{CE}(\mathbf{Y}, \mathbf{d}, f_\theta) = -\frac{1}{K} \sum_{k=1}^{K} log(p_{y_k}) \tag{2.3}$$

A series of neural methods have been proposed for the NLG task, including HLSTM (Wen et al., 2015a), SCLSTM (Wen et al., 2015b), Enc-Dec (Wen et al., 2015c) and RAL-STM (Tran and Nguyen, 2017). The goal of low-resource NLG is to fine-tune a trained NLG model on new NLG tasks (e.g., new domains) with a small number of training examples. (Wen et al., 2016) proposed a "data counterfeiting" method to augment the low-resource training data in the new task without modifying the model or training procedure. (Tseng et al., 2018) proposed a semantically-conditioned variational autoencoder (SCVAE) learn domain-invariant representations feeding to SCLSTM. They showed that it improves SCLSTM in low-resource settings. (Tran and Nguyen, 2018b) adopted the same idea as in (Tseng et al., 2018). They used two conditional variational autoencoders to encode the sentence and the DA into two separate latent vectors, which are fed together to the decoder RALSTM (Tran and Nguyen, 2017). They later designed two domain adaptation critics with an adversarial training algorithm (Tran and Nguyen, 2018a) to learn an indistinguishable latent representation of the source and the target domain to better generalize to the target domain. Different from these model-based approaches, we directly tackle the optimization issue from a meta-learning perspective.

2.3 Methodology - Meta-NLG

In this section, we first describe the objective of fine-tuning an NLG model on a low-resource NLG task. Then, we describe how our Meta-NLG algorithm encapsulates this objective into *Meta NLG tasks* and into the meta optimization algorithm to learn better from source domains for the low-resource NLG task in new domains.

2.3.1 Fine-tune a NLG Model

Suppose f_θ is the base NLG model parameterized by θ, and we have an initial $\theta^{\mathbf{s}}$ pre-trained with DA-utterance pairs $\mathcal{D}_{\mathbf{s}} = \{(\mathbf{d}_j, \mathbf{Y}_j)\}_{j\in\mathbf{s}}$ from a set $\mathbf{s}$ of high-resource source tasks. When we adapt f_θ to a low-resource task t with DA-utterance pairs $\mathcal{D}_t = (\mathbf{d}_t, \mathbf{Y}_t)$, the fine-tuning process on $\mathcal{D}_t$ can be formulated as follows:

$$\begin{aligned}\theta^* &= Adapt(\mathcal{D}_t, \theta = \theta^{\mathbf{s}}) = \arg\min_\theta \mathcal{L}_{\mathcal{D}_t}(f_\theta) \\ &= \arg\min_\theta \sum_{(\mathbf{d}_t, \mathbf{Y}_t)\in\mathcal{D}_t} \mathcal{L}_{CE}(\mathbf{Y}_t, \mathbf{d}_t, f_\theta)\end{aligned} \tag{2.4}$$

The parameter $\theta^{\mathbf{s}}$ will be used for initialization, and the model is further updated by new observations $\mathcal{D}_t$. The size of $\mathcal{D}_t$ in low-resource NLG tasks is very small due to the high annotation cost. Therefore, a good initialization parameter $\theta^{\mathbf{s}}$ learned from high-resource source tasks is crucial for the adaptation performance on new low-resource NLG tasks.

2.3.2 Meta NLG Tasks

To learn a $\theta^{\mathbf{s}}$ that can be easily fine-tuned on new low-resource NLG tasks, the idea of our Meta-NLG algorithm is to repeatedly simulate auxiliary *Meta NLG tasks* from $\mathcal{D}_{\mathbf{s}}$ to mimic the fine-tuning process in Eq.(2.4). Then, we treat each *Meta NLG task* as a single meta training sample/episode and utilize the meta optimization objective in the next section to directly learn from them.

Therefore, the first step is to construct a set of auxiliary *Meta NLG tasks* to simulate the low-resource fine-tuning process. We construct a *Meta NLG task* $\mathcal{T}_i$ by:

$$\mathcal{T}_i = (\mathcal{D}_{\mathcal{T}_i}, \mathcal{D}'_{\mathcal{T}_i}) \tag{2.5}$$

$\mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}'_{\mathcal{T}_i}$ of each $\mathcal{T}_i$ are two independent subsets of DA-utterance pairs from high-resource source data $\mathcal{D}_{\mathbf{s}}$. $\mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}'_{\mathcal{T}_i}$ correspond to meta-train (support) and meta-test (query) sets of a typical meta-learning or few-shot learning setup, and $\mathcal{T}_i$ is often referred to as a training episode. This meta setup with both $\mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}'_{\mathcal{T}_i}$ in one *Meta NLG task* allows our Meta-NLG algorithm to directly learn from different *Meta NLG tasks*. The usage of them will be elaborated later. *Meta NLG tasks* are constructed with two additional principles:

Task modality consistency. To generalize to new NLG tasks, *Meta NLG tasks* follow the same *modality* as the target task. For example, if our target task is to adapt to DA-utterance pairs in a new domain, then DA-utterance pairs in each $\mathcal{T}_i$ are sampled from the same source domain. We also consider adapting to new DA intents in later experiments. In this case, DA-utterance pairs in each $\mathcal{T}_i$ have the same DA intent. This setting merges the goal of task generalization.

Low-resource adaptation. To simulate the process of adapting to a low-resource NLG task, the sizes of both subsets $\mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}'_{\mathcal{T}_i}$, especially $\mathcal{D}_{\mathcal{T}_i}$, are set small. Therefore, when the model is updated on $\mathcal{D}_{\mathcal{T}_i}$ as a part of the later meta-learning steps, it only sees a small number of samples in that task. This setup embeds the goal of low-resource adaptation.

2.3.3 Meta Training Objective

With the *Meta NLG tasks* defined above, we formulate the meta-learning objective of Meta-NLG as below:

$$\begin{aligned} \theta^{Meta} &= MetaLearn(\mathcal{T}_i) \\ &= \arg\min_{\theta} \mathbb{E}_i \mathbb{E}_{\mathcal{D}_{\mathcal{T}_i}, \mathcal{D}'_{\mathcal{T}_i}} \mathcal{L}_{\mathcal{D}'_{\mathcal{T}_i}}(f_{\theta'_i}) \end{aligned} \tag{2.6}$$

$$\theta'_i = Adapt(\mathcal{D}_{\mathcal{T}_i}, \theta) = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_{\mathcal{T}_i}}(f_\theta) \tag{2.7}$$

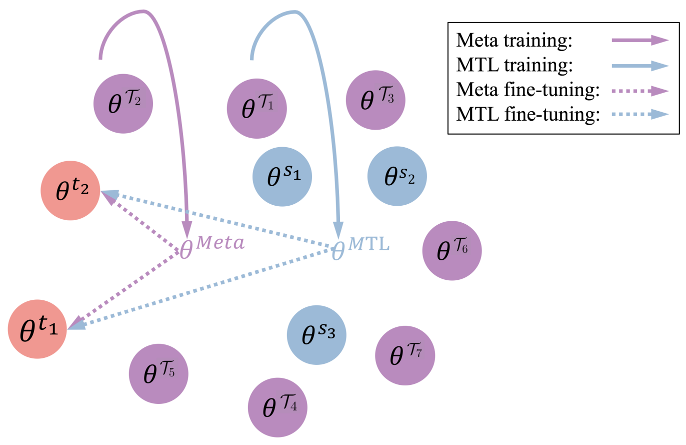


Figure 2.1 – Comparing Meta-Learning to Multi-task Learning: θ^{Meta} meta-learned from auxiliary *Meta NLG tasks* can be fine-tuned easier than θ^{MTL} to some new low-resource tasks, e.g, t_1 and t_2.

The optimization for each Meta NLG task $\mathcal{T}_i$ is computed on $\mathcal{D}'_{\mathcal{T}_i}$ referring to $\mathcal{D}_{\mathcal{T}_i}$. **Firstly**, the model parameter θ to be optimized is updated on $\mathcal{D}_{\mathcal{T}_i}$ by Eq.(2.7). This step mimics the process when f_θ is adapted to a new low-resource NLG task $\mathcal{T}_i$ with low-resource observations $\mathcal{D}_{\mathcal{T}_i}$. We need to note that Eq.(2.7) is an intermediate step, and it only provides an adapted parameter (θ'_i) to our base model f to be optimized in each iteration. **Afterwards**, the base model parameterized by the updated parameter (θ'_i) is optimized on $\mathcal{D}'_{\mathcal{T}_i}$ using the meta objective in Eq.(2.6). This meta-learning optimization objective directly optimizes the model towards generalizing to new low-resource NLG tasks by simulating the process repeatedly with *Meta NLG tasks* in Eq.(2.6).

The optimization of Eq.(2.6) can be derived in Eq.(2.8) with aggregating K inner updates before updating θ. It involves a standard first-order gradient $\nabla_{\theta'_i}\mathcal{L}_{\mathcal{D}'_{\mathcal{T}_i}}(f_{\theta'_i})$ as well as a gradient through another gradient $\nabla_\theta(\theta'_i)$. Previous study (Finn et al., 2017) shows that the second term can be approximated for computation efficiency with a marginal performance drop. In our case, we still use the exact optimization in Eq.(2.8) as we do not encounter any computation difficulties even on the largest NLG dataset so far. The second-order gradient is computed by a Hessian matrix H.

$$\begin{aligned}
\theta \leftarrow \theta &- \beta \sum_{i=1}^{K} \nabla_\theta \mathcal{L}_{\mathcal{D}'_{\mathcal{T}_i}}(f_{\theta'_i}) \\
&= \theta - \beta \sum_{i=1}^{K} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{D}'_{\mathcal{T}_i}}(f_{\theta'_i}) \cdot \nabla_\theta(\theta'_i) \\
&= \theta - \beta \sum_{i=1}^{K} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{D}'_{\mathcal{T}_i}}(f_{\theta'_i}) \cdot \nabla_\theta(\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_{\mathcal{T}_i}}(f_\theta)) \\
&= \theta - \beta \sum_{i=1}^{K} \nabla_{\theta'_i} \mathcal{L}_{\mathcal{D}'_{\mathcal{T}_i}}(f_{\theta'_i}) \cdot (I - \alpha H_\theta(\mathcal{L}_{\mathcal{D}_{\mathcal{T}_i}}(f_\theta)))
\end{aligned} \tag{2.8}$$

Algorithm 1 Meta-NLG($f_\theta, \theta_0, \mathcal{D}_\mathbf{s}, \alpha, \beta$)

Input: Base NLG model: f_θ, Initial parameter: θ_0, Training data: $\mathcal{D}_\mathbf{s}$, Inner learning rate: α, Meta learning rate: β, Number of inner updates before updating θ: K

Output: θ^{Meta}

```
1: Initialize $\theta = \theta_0$
2: while $\theta$ not converge do
3:     Simulate a batch of Meta NLG tasks $\{\mathcal{T}_i = (\mathcal{D}_{\mathcal{T}_i}, \mathcal{D}'_{\mathcal{T}_i})\}_{i=1}^K$
4:     for $i = 1...K$ do
5:         Compute $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_{\mathcal{T}_i}}(f_\theta)$ in Eq.(2.7)
6:     end for
7:     Meta update $\theta \leftarrow \theta - \beta \sum_{i=1}^K \nabla_\theta \mathcal{L}_{\mathcal{D}'_{\mathcal{T}_i}}(f_{\theta'_i})$ in Eq.(2.8)
8: end while
9: $\theta^{Meta} \leftarrow \theta$
```

To better understand the meta objective, we compare a standard multi-task learning (MTL) objective $\theta^{MTL} = \arg\max_\theta \mathbb{E}_j \mathcal{L}_{\mathcal{D}_{s_j}}(f_\theta)$ that learns from high-resource NLG tasks s_j without explicitly learning to adapt to low-resource NLG tasks. Figure 2.1 visually illustrates the differences with three high-resource source tasks $\{s_1, s_2, s_3\}$ with optimal parameters $\{\theta^{s_1}, \theta^{s_2}, \theta^{s_3}\}$ for each task. θ^{MTL} is learned from individual DA-utterance pairs in $\{\mathcal{D}_{s_1}, \mathcal{D}_{s_2}, \mathcal{D}_{s_3}\}$, while Meta-NLG repeatedly constructs auxiliary *Meta NLG tasks* $\{\mathcal{T}_1, ..., \mathcal{T}_7\}$ from $\{\mathcal{D}_{s_1}, \mathcal{D}_{s_2}, \mathcal{D}_{s_3}\}$ and learns θ^{Meta} from them. As a result, θ^{Meta} is closer to θ^{t_1} and θ^{t_2} (the optimal parameters of some new low-resource tasks, e.g, t_1 and t_2) than θ^{MTL}. Algorithm 1 illustrates the process to learn θ^{Meta} from $\mathcal{D}_\mathbf{s}$. We note that batches are at the level of *Meta NLG tasks*, not DA-utterances pairs. Fine-tuning Meta-NLG on a new low-resource NLG task with annotated DA-utterance pairs $\mathcal{D}_t$ uses the same algorithm parameterized by $(f_\theta, \theta^{Meta}, \mathcal{D}_t, \alpha, \beta, K)$.

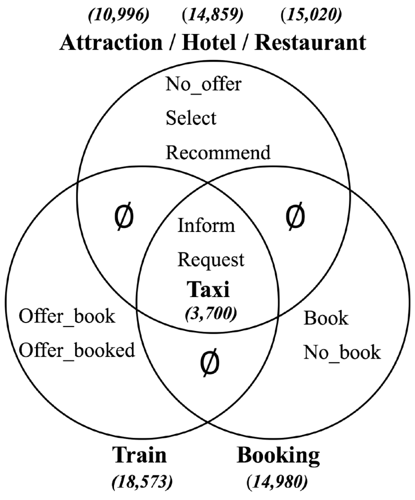
2.4 Evaluation

2.4.1 Baselines and Experiment Settings

We utilize the well-recognized semantically conditioned LSTM (SCLSTM Wen et al. (2015b)) as the base model f_θ. We use the default setting of hyperparameters (n_layer = 1, hidden_size = 100, dropout = 0.25, clip = 0.5, beam_width = 5). We implementMeta-NLG based on the PyTorch SCLSTM implementation from (Budzianowski et al., 2018). As Meta-NLG is model-agnostic, it is applicable to many other NLG models, and evaluation using other NLG models is left as future work.

We include different model settings as baseline:

- **Scratch-NLG:** Train f_θ with only low-resource target task data, ignoring all high-resource source task data.



Attraction	Addr, Area , Choice, **Fee**, Name, **Open**, Phone, Post , Price, Type
Hotel	Addr, Area, Choice, **Internet**, Name, **Parking**, Phone, Post, Price, Ref, **Stars**, Type
Restaurant	Addr, Area, Choice, **Food**, Name, Phone, Post, Price, Ref
Train	Arrive, Choice, Day, Depart, Dest, **Id**, Leave, People, Ref, **Ticket**, Time
Booking	Day, Name, **People**, Ref, **Stay**, Time
Taxi	Arrive, **Car**, Depart, Dest, Leave, Phone

Figure 2.2 – MultiWOZ dataset visualization. **Left**: DA intent visualization in different domains, and the number of utterances in each domain is indicated in bracket. **Right**: Slots in each domain, with domain-specific slots in bold.

- **MTL-NLG:** Train f_θ using a multi-task learning paradigm on source task data, then fine-tune on the low-resource target task.
- **Zero-NLG:** Train f_θ using multi-task learning (MTL) with source task data, then directly test on a target task without a fine-tuning step. This corresponds to a zero-shot learning scenario.
- **Supervised-NLG:** Train f_θ using MTL with full access to high-resource data from both source and target tasks. Its performance serves as an upper bound using multi-task learning without the low-resource restriction.
- **Meta-NLG (proposed):** Train f_θ using Algorithm 1 on source task data, then fine-tune on the low-resource target task.

For Meta-NLG, we set batch size to 50, $\alpha = 0.1$, and $\beta = 0.001$. A single inner gradient update ($K = 1$) is used per meta update with Adam (Kingma and Ba, 2015). The size of a *Meta NLG task* is set to 400 with 200 samples assigned to $\mathcal{D}_{\mathcal{T}_i}$ and $\mathcal{D}'_{\mathcal{T}_i}$. The maximum number of epochs is set to 100 during training and fine-tuning, and early-stop is conducted on a small validation set with size 200. The model is then evaluated on other DA-utterance pairs in the target task.

As in earlier NLG researches, we use the BLEU-4 score (Papineni et al., 2002) and the slot error rate (SER) as evaluation metrics. SER is computed by the ratio of the sum of the number of missing and redundant slots in a generated utterance divided by the total number of slots in the DA. We randomly sample the target low-resource task five times for each experiment and reported the average score.

	Target Domain = **Attraction**					
	Supervised-NLG			**Zero-NLG**		
	BLEU-4		SER	BLEU-4		SER
	0.5587		3.05%	0.2970		11.56%
	Adapt 1000		Adapt 500		Adapt 200	
	BLEU-4	SER	BLEU-4	SER	BLEU-4	SER
Scratch-NLG	0.5102	21.84%	0.4504	36.50%	0.4089	41.83%
MTL-NLG	0.5443	13.04%	0.5324	14.34%	0.4912	23.20%
Meta-NLG	**0.5667**	**2.26%**	**0.5662**	**2.97%**	**0.5641**	**4.30%**

Table 2.1 – Results for near-domain ("Attraction") adaption with different adaptation sizes. Bold numbers highlight the best results except Supervised-NLG.

	Target Domain = **Hotel**					
	Supervised-NLG			**Zero-NLG**		
	BLEU-4		SER	BLEU-4		SER
	0.4393		1.82%	0.2514		13.40%
	Adapt 1000		Adapt 500		Adapt 200	
	BLEU-4	SER	BLEU-4	SER	BLEU-4	SER
Scratch-NLG	0.3857	18.75%	0.3529	28.18%	0.2910	40.86%
MTL-NLG	0.4128	9.93%	0.3802	22.07%	0.3419	31.04%
Meta-NLG	**0.4436**	**1.92%**	**0.4365**	**2.63%**	**0.4418**	**2.19%**

Table 2.2 – Results for near-domain ("Hotel") adaption with different adaptation sizes. Bold numbers highlight the best results except Supervised-NLG.

We use a large-scale multi-domain dialog dataset (MultiWOZ, Budzianowski et al. (2018)). The version extracted for NLG can be found at https://github.com/andy194673/nlg-sclstm-multiwoz. It is a proper benchmark for evaluating NLG components due to its domain complexity and rich linguistic variations. The average utterance length is 15.12, and almost 60% of utterances have more than one dialogue act intents or domains. 69,607 annotated utterances are used, with 55,026, 7,291, 7,290 for training, validation, and testing respectively. A visualization of DA intents in different domains are given in Figure 2.2 (Left), and slots in different domains are summarized in Figure 2.2 (Right).

2.4.2 Domain Adaptation Results

In this section, we test when an NLG model is adapted to various low-resource language domains. The experiment follows a *leave-one-out* setup by leaving one target domain for adaptation, while using the remainder domains for training. A target domain is a *near-domain* if it only contains domain-specific slots compared to the remainder domains. In contrast, a target domain containing both domain-specific DA intents and slots is considered as a *far-domain.* According to Figure 2.2, "Attraction", "Hotel", "Restaurant", and "Taxi", are near-domains, while "Booking" and "Train" are far-domains compared to

	Target Domain = **Booking**					
	Supervised-NLG			**Zero-NLG**		
	BLEU-4		SER	BLEU-4		SER
	0.6750		3.67%	0.3578		12.55%
	Adapt 1000		Adapt 500		Adapt 200	
	BLEU-4	SER	BLEU-4	SER	BLEU-4	SER
Scratch-NLG	0.6327	24.63%	0.6267	37.96%	0.5787	46.67%
MTL-NLG	0.6347	14.55%	0.6391	14.90%	0.6171	17.19%
Meta-NLG	**0.6782**	**7.65%**	**0.6492**	**9.08%**	**0.6402**	**12.23%**

Table 2.3 – Results for far-domain ("Booking") adaption with different adaptation sizes. Bold numbers highlight the best results except Supervised-NLG.

	Target Domain = **Train**					
	Supervised-NLG			**Zero-NLG**		
	BLEU-4		SER	BLEU-4		SER
	0.6877		2.96%	0.3243		41.48%
	Adapt 1000		Adapt 500		Adapt 200	
	BLEU-4	SER	BLEU-4	SER	BLEU-4	SER
Scratch-NLG	0.6236	16.73%	0.5825	27.61%	0.4892	44.92%
MTL-NLG	0.6322	14.63%	0.5987	25.38%	0.5248	40.35%
Meta-NLG	**0.6755**	**7.13%**	**0.6373**	**17.31%**	**0.6160**	**23.33%**

Table 2.4 – Results for far-domain ("Train") adaption with different adaptation sizes. Bold numbers highlight the best results except Supervised-NLG.

the remaining domains. Adapting to near-domains requires to capture unseen slots while adapting to far-domains additionally requires to learn new language patterns. *Adaptation size* is the number of DA-utterance pairs in the target domain to fine-tune the NLG model. To test different low-resource degrees, we consider different *adaptation sizes* (1,000, 500, 200).

Near-domain adaptation. Results of adapting to two near-domains ("Attraction" and "Hotel") are presented in Table 2.1 and 2.2. Other two near-domains ("Restaurant", and "Taxi") are simpler, therefore, they are not included. Several observations can be noted. **First**, using only source or target domain samples does not produce competitive performance. Using only source domain samples (Zero-NLG) performs the worst. It obtains very low BLEU-4 scores, indicating that the sentences generated do not match the linguistic patterns in the target domain. Using only low-resource target domain samples (Scratch-NLG) performs slightly better, yet still much worse than MTL-NLG and Meta-NLG. **Second**, Meta-NLG shows a very strong performance for this near-domain adaptation setting. It consistently outperforms MTL-NLG and other methods with very remarkable margins in different metrics and adaptation sizes. More importantly, it even works better than Supervised-NLG which is trained on high-resource samples

	Book		Recommend	
	BLEU-4	SER	BLEU-4	SER
Scratch-NLG	0.7689	21.63%	0.3878	24.62%
MTL-NLG	0.7968	9.92%	0.3964	14.60%
Meta-NLG	**0.8217**	**4.65%**	**0.4445**	**3.08%**

Table 2.5 – Results for adapting to new DA intent "Book" and "Recommend" with adaptation size 500.

in the target domain. **Third**, Meta-NLG is particularly strong in performance when the adaptation size is small. As the adaptation size decreases from 1,000 to 200, the performance of Scratch-NLG and MTL-NLG drops quickly, while Meta-NLG performs stably well. Both BLEU-4 and SER even increase in the "Hotel" domain when the adaptation size decreases from 500 to 200.

Far-domain adaptation. Results of adapting to two far-domains ("Booking" and "Train") are presented in Table 2.3 and 2.4. Again, we can see that Meta-NLG shows very strong performance on both far-domains with different adaptation sizes. Similar observations can be made as in the previous near-domain adaptation experiments. Because far-domain adaptation is very challenging, Meta-NLG does not outperform Supervised-NLG, and the performance of Meta-NLG drops more obviously as the adaptation size decreases. Noticeably, "Train" is more difficult than "Booking" as the former contains more slots, some of which can only be inferred from the smallest "Taxi" domain. The improvement margin of Meta-NLG over MTL-NLG and other methods is larger on the more difficult "Train" domain than on the "Booking" domain.

2.4.3 DA Intent Adaptation Results

It is also important for a task-oriented dialog system to adapt to new functions, namely, supporting new dialog acts that the system has never observed before. To test this ability, we leave certain DA intents out for adaptation in a low-resource setting. We choose "Recommend", "Book" as target DA intents, and we mimic the situation that a dialog system needs to add a new function to make *recommendations* or *bookings* for customers with a small number of annotated DA-utterance pairs. As presented in Table 2.5, results show that Meta-NLG significantly outperforms other baselines.

2.4.4 Adaptation Curve Analysis

After seeing the superior performance of Meta-NLG, we move to an insightful analysis of the results. To further investigate the adaptation process, we present in Figure 2.3 the performance curves of MTL-NLG and Meta-NLG as fine-tuning epoch proceeds on the

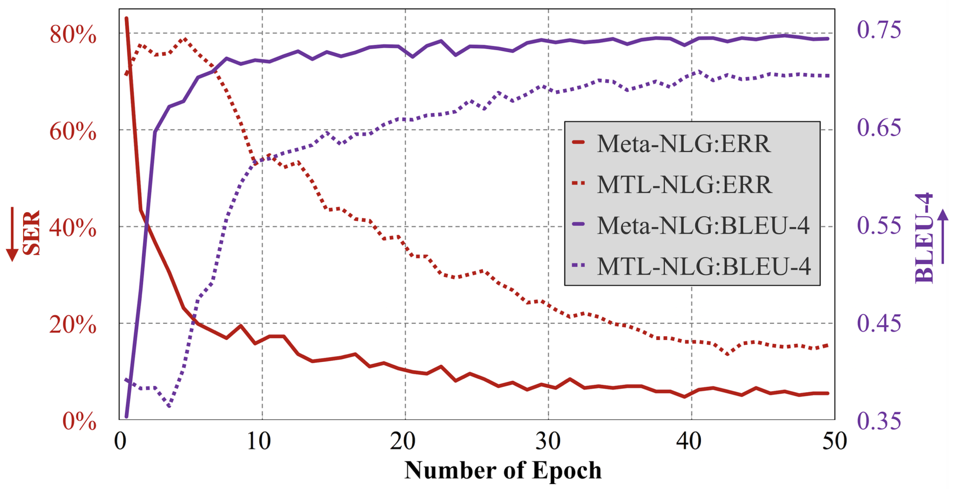


Figure 2.3 – SERs (red) and BLEU-4 (purple) scores of Meta-NLG and MTL-NLG on the validation set during model fine-tuning on the target low-resource domain (Train) with adaptation size 1000.

most challenging "Train" domain. The effect of meta-learning for low-resource NLG can be observed by comparing the two solid curves against the corresponding dashed curves. **First**, Meta-NLG adapts *faster* than MTL-NLG. We can see that the SER of Meta-NLG (red-solid) decreases much more rapidly than that of MTL-NLG (red-dashed), and the BLEU-4 score of Meta-NLG (purple-solid) also increases more quickly. The optimal BLEU-4 and SER that MTL-NLG converges to can be obtained by Meta-NLG within 10 epochs. **Second**, Meta-NLG adapts *better* than MTL-NLG. As it can be seen, Meta-NLG achieves a much lower SER and a higher BLEU-4 score when it converges, indicating that it found a better θ of the base NLG model to generalize to the low-resource target domain.

2.4.5 Human Evaluation Results

To better evaluate the quality of the generated utterances, we conduct a human evaluation.

Metrics. Given a DA and a reference utterance in a low-resource target domain with adaptation size 500, two responses generated by Meta-NLG and MTL-NLG are presented to three human annotators to score each of them in terms of *informativeness* and *naturalness* (rating out of 3), and also indicate their *pairwise preferences* (Win-Tie-Lose) on Meta-NLG against MTL-NLG. *Informativeness* is defined as whether the generated utterance captures all the information, including multiple slots and probably multiple DA intents, specified in the DA. *Naturalness* measures whether the utterance is plausibly generated by a human.

	Attraction		Hotel		Booking		Train	
	inf.	nat.	inf.	nat.	inf.	nat.	inf.	nat.
Meta-NLG	2.91	2.90	2.90	2.89	2.84	2.91	2.73	2.93
MTL-NLG	2.70	2.87	2.57	2.83	2.65	2.86	2.47	2.83

Table 2.6 – Manual evaluation scores of *informativeness* (inf.), and *naturalness* (nat.) on four target low-resource domains.

	Attraction	Hotel	Booking	Train
Win	50.2%	53.3%	40.1%	47.2%
Tie	42.8%	42.3%	46.2%	40.5%
Lose	7.0%	4.4%	13.7%	12.3%

Table 2.7 – Pairwise preferences (Meta-NLG vs. MTL-NLG) on four target low-resource domains.

Annotation statistics. Cases with identical utterances generated by two models are filtered out. We obtain in total 600 annotations on each individual metric for each target domain. We calculate the Fleiss' kappa (Fleiss, 1971) to measure inter-rater consistency. The overall Fleiss' kappa values for *informativeness* and *naturalness* are 0.475 and 0.562, indicating "Moderate Agreement", and 0.637 for *pairwise preferences*, indicating "Substantial Agreement".

Results. Scores of *informativeness* and *naturalness* are presented in Table 2.6. Meta-NLG outscores MTL-NLG in terms of both metrics on all four domains. Overall, Meta-NLG received *significantly* (two-tailed t-test, $p < 0.0005$) higher scores than MTL-NLG. Results for *pairwise preferences* are summarized in Table 2.7. Even though there are a certain amount of cases where the utterances generated by different models are nearly the same (Tie) to annotators, Meta-NLG is overall *significantly* preferred over MTL-NLG (two-tailed t-test, $p < 0.0001$) across different target domains.

2.4.6 Case Study

Table 2.8 shows two examples in the "Train" domain. The first sample shows that MTL-NLG fails to generate the domain-specific slot "Ticket"; instead, it mistakenly generates slots ("Leave" and "Arrive") that are frequently observed in the low-resource adaptation set. In the second example, MTL-NLG failed to generate the domain-specific slot 'Id" and another rare slot "Dest", while Meta-NLG succeeded both.

Inform (**Ticket**†=**17.60 pounds**, **Time**=**79 minutes**); Offer_book*(None)	
Reference	*The travel time is [**79 minutes**] and the cost is [**17.60 pounds**], shall I book for you?*
MTL-NLG	*there is a train that leaves at [**slot-train-leave**] and arrives at [**slot-train-arrive**]. would you like me to book it for you?* [**missed: Ticket**†, **Time**; **redundant: Leave, Arrive**]
Meta-NLG	*the travel time is [**79 minutes**] and the price is [**17.60 pounds**]. would you like me to book it for you?* [**correct**]
Inform(**Arrive**=**7:52**, **Id**†=**TR9641**, **Dest**‡=**cambridge** , **Depart**‡=**the airport**, **Leave** =**7:24**)	
Reference	*[**TR9641**] leaves [**the airport**] at [7:24] and arrives in [**cambridge**] at [7:52].*
MTL-NLG	*i have a train that leaves [**the airport**] at [7:24] and arrives by [7:52].* [**missed: Id**†, **Dest**‡]
Meta-NLG	*[**TR9641**] leaves [**the airport**] at [7:24] and arrives in [**cambridge**] at [7:52].* [**correct**]

Table 2.8 – Sampled generated sentences when considering "Train" as the target domain with adaptation size 500. * indicates a *domain-specific DA intent*, † indicates a *domain-specific slot*, and ‡ indicates a *rare slot* that can only be inferred from the smallest "Taxi" domain. Redundant and missing slots are colored in orange and blue respectively.

2.5 Chapter Summary

In this chapter, we propose a generalized optimization-based meta-learning approach Meta-NLG for the low-resource NLG task in task-oriented dialog systems to learn new domains or new DA intents with limited annotations. Meta-NLG utilizes *Meta NLG tasks* and a meta-learning optimization procedure based on MAML. Extensive experiments on a large-scale benchmark dataset (MultiWOZ) show that Meta-NLG significantly outperforms other training procedures, indicating that it adapts fast and well to new low-resource settings. Our work may inspire researchers to use similar optimization techniques for building more scalable ToD systems with high data efficiency.

3 Self-training Improves Data Efficiency in ToD Systems

3.1 Introduction

Large-scale pre-trained language models, such as BERT (Devlin et al., 2019a), ULMFiT (Howard and Ruder, 2018), GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019), and GPT-3 (Brown et al., 2020), have shown great few-shot or zero-shot learning abilities in various NLP tasks with the help of task-agnostic language knowledge learned via pre-training tasks. Pre-training can be seen as a semi-supervised technique, in which models are first trained on an auxiliary task to learn task-agnostic language knowledge, such as language modeling, followed by learning the task of interest. In task-oriented dialog (ToD) systems, the labeling cost is very high such that the size of well-labeled data is often small. Therefore, few-shot learning in ToD is essential and valuable in many practical applications. Several attempts (Peng et al., 2020b,a; Wu et al., 2020) have been proposed to leverage large-scale pre-trained language models to improve few-shot learning in ToD. Specifically, a model pre-trained on general text corpora is further trained on public ToD datasets.

Although labeled data is often small, a practical ToD system de facto has many unlabeled dialog data. Therefore, utilizing unlabeled data to improve a ToD system is practically important. In this chapter, we take a semi-supervised self-training (ST) perspective to iteratively train a better *Student* model using unlabeled data (III, 1965; Yarowsky, 1995). ST has been successfully applied to a variety of tasks, including image classification (Yalniz et al., 2019; Xie et al., 2020; Zoph et al., 2020), automatic speech classification (Synnaeve et al., 2019; Kahn et al., 2020; Park et al., 2020; Likhomanenko et al., 2020), sequence generation (He et al., 2020), and natural language understanding (Du et al., 2020).

We are going to study this research question: *can self-training provide complementary benefits on top of the strong pre-training models for few-shot learning in ToD?* Recently,

This chapter is based on an ongoing work under review.

Xie et al. (2020); Zoph et al. (2020) studied a similar question in the context of image classification, showing that ST effectively refines pre-training models. Du et al. (2020) also recently showed the benefit of ST over pre-training for general natural language understanding. Nevertheless, their main proposal is to crawl a large amount of similar unlabeled data from the web.

In this chapter, we propose a self-training approach based on iterative pseudo-labeling (Lee, 2013). It first trains a *Teacher* on the labeled samples. The *Teacher* then iteratively generates pseudo-labels for the most confident subset of unlabeled samples to train a better *Student*. To train a more robust *Student* during self-training, we propose a data augmentation technique called GradAug. GradAug first "masks" a fraction of tokens of a dialog input. Then, it reconstructs the corrupted text with a pre-trained masked language model of BERT. Different from Ng et al. (2020), the probability of masking a token is conditioned on the gradient of the corresponding token embedding w.r.t. the downstream task. In this way, GradAug prevents replacing tokens that are critical for a downstream task.

The main contribution of this chapter is three-fold:

- This is the first attempt to study the effect of self-training on top of existing strong pre-trained models for ToD in few-shot learning scenarios.
- We propose a self-training method to gradually train a stronger *Student* by iteratively labeling the most confident unlabeled data and a new text augmentation technique (GradAug).
- We conduct extensive experiments on four downstream tasks in ToD, including intent classification, dialog state tracking, dialog act prediction, and response selection. Empirical results demonstrate that self-training *consistently* improves state-of-the-art pre-trained models (BERT, ToD-BERT (Wu et al., 2020)).

3.2 Related Work

3.2.1 Pre-training for ToD Systems

Budzianowski and Vulic (2019) first applied GPT-2 to train a response generation model by taking the system belief state, database entries, and last dialog turn as input. Henderson et al. (2019) pre-trained a response selection model for ToD by first pre-training on general-domain conversational corpora (Reddit). Ham et al. (2020) trained the pre-trained GPT-2 for dialog state tracking and response generation on MultiWOZ (Budzianowski et al., 2018). Hosseini-Asl et al. (2020) proposed SimpleToD to train the pre-trained GPT-2 on three different sub-tasks (dialog state tracking, dialog act prediction, and response generation) of ToD as a sequence prediction problem.

Recent studies have shown that large-scale pre-trained language models are good few-shot learners (Brown et al., 2020). Several studies have also confirmed these findings for ToD. Peng et al. (2020b) proposed further to train a GPT-2 on public ToD corpora to improve few-shot learning for the task of generating responses conditioned on a semantic representation. Peng et al. (2020a) utilized GPT-2 for end-to-end response generation from dialog contexts in a few-shot learning scenario. Wu et al. (2020) further trained a BERT model on multiple ToD corpora to improve few-shot learning performance on four different downstream tasks.

3.2.2 Self-training

The first focus of self-training is designing better policies to label unlabeled samples. Zhang and Zhou (2011) evaluated the confidence via a statistic-based data editing technique. Lee (2013) designed an annealing function that gradually increases the loss of labeled samples during training to avoid poor local minima. Amiri (2019) utilized a Leitner queue (Dempster, 1989) to put confident samples in the front of the queue gradually. Niu et al. (2020) selected the most confident samples with prediction loss below some threshold. Kumar et al. (2010); Ma et al. (2017); Li et al. (2019); Mukherjee and Awadallah (2020) proposed to learn different sampling weights for unlabeled data to control the selection process. Weights are learned asynchronously with the model parameters using different metrics. For example, Ma et al. (2017); Kumar et al. (2010) considers the loss on the validation set; Mukherjee and Awadallah (2020) selects samples that maximize the information gain about the model parameters; Ren et al. (2020) computes the influence on current parameters when changing the sampling weights. Reinforcement learning (RL) methods (Chen et al., 2018; Wu et al., 2018; Ye et al., 2020) designed an additional Q-agent as the sample selector. Nevertheless, methods using learnable weights or RL provide marginal benefits compared to the elevated optimization cost. Designing new sample selection schemes is not our primary focus; we will go for a simple and effective pipeline described in Section 3.4.1. Specialized explorations on this topic are orthogonal to ours.

The second focus of self-training is to improve the robustness of the *Student* model trained from potentially noisy pseudo-labeled samples. Data augmentation techniques are widely used. In computer vision, recent works demonstrated the benefit of different stochastic augmentation tricks, including input transformations (Laine and Aila, 2017; Xie et al., 2020; Zoph et al., 2020), dropout (Laine and Aila, 2017; Xie et al., 2020; Zoph et al., 2020), adversarial samples (Miyato et al., 2019), and Mixup (Berthelot et al., 2019, 2020). Text augmentation is very challenging because of the complex syntactic and semantic structures. Miyato et al. (2017) utilized adversarial training to apply perturbations to word embeddings. Wei and Zou (2019) proposed EDA using basic synonym replacement, random insertion, swap, and deletion. Kumar et al. (2019) proposed to maximize a monotone sub-modular function to obtain diverse paraphrases.

Xie et al. (2019) proposed UDA applying back-translation (Edunov et al., 2018) and word replacement using a Tf-Idf metric. He et al. (2020) studied the effect of dropout compared to back-translation during self-training for the neural sequence generation task. Chen et al. (2020a) proposed MixText that utilizes Manifold Mixup (Verma et al., 2019) to interpolate hidden layers corresponding to semantic representations of BERT. Ng et al. (2020) proposed SSMBA utilizing the masked language model of BERT to replace words. In experiments, we compare the proposed GradAug technique with state-of-the-art text augmentation methods.

3.3 Background of Using Pre-trained Models for Downstream Tasks in ToD

In this section, we first briefly overview the pipeline of utilizing large-scale pre-trained models for four common downstream tasks (intent classification, dialog state tracking, dialog act prediction, and response selection) in ToD. We denote the input and label of different downstream tasks as x and y, and a prediction model is denoted as $\hat{y}_x = F(x)$. F can often be decomposed into two parts. The first part is a *feature extractor* $\boldsymbol{h} = A(x) \in \mathbb{R}^l$ which computes a hidden representation $\boldsymbol{h}$ of x, and the second part is an *output network* for prediction. Large-scale pre-trained language models serve as *feature extractor* A to compute a hidden representation for an input. For example, we use the [CLS] embedding of BERT as the hidden representation $\boldsymbol{h}$ when BERT is adopted as A. Different *output networks* are designed for different downstream tasks, and the details following ToD-BERT (Wu et al., 2020) are described below.

Intent classification. This is a multi-class classification problem to predict the single intent label y of an input utterance x. The model computes the probability over I possible intents as:

$$\boldsymbol{p}_{int} = Softmax(W_1 \cdot A(x)) \in \mathbb{R}^I, \tag{3.1}$$

where $W_1 \in \mathbb{R}^{I \times l}$ is a trainable weight matrix, and the model is optimized by the standard cross-entropy loss compared to the ground truth.

Dialog state tracking. It is a multi-class classification problem based on a predefined ontology. Unlike intent classification, the dialog history (a sequence of utterances) is used as the input x. For each (domain, slot) pair, the model predicts a score over all potential slot values. For the i-th slot value v_i^j of the j-th pair, the cosine similarity score compared to the input x is computed as follows:

$$s_i^j = Cosine(G_j(A(x)), A(v_i^j)) \in \mathbb{R}^1, \tag{3.2}$$

where G_j is the slot projection layer of the j-th pair, and the number of layers $|G|$ equals the number of (domain, slot) pairs. The model is trained with the cross-entropy loss summed over all the pairs.

Dialog act prediction. This is a multi-label classification problem to predict the dialog act (DA) intents for the next system response. The model takes a dialog history as input x and predicts a Bernoulli outcome for each possible DA intent as:

$$\boldsymbol{a} = Sigmoid(W_2 \cdot A(x)) \in \mathbb{R}^N, \tag{3.3}$$

where $W_2 \in \mathbb{R}^{N \times l}$ is a trainable weight matrix, and N is the number of possible DA intents. Values in $\boldsymbol{a}$ are between $[0, 1]$, and the model is optimized by a binary cross-entropy loss w.r.t. the ground truth. A threshold of 0.5 is applied during inference.

Response selection. This task predicts the most relevant system response from a candidate pool. A dual-encoder model (Henderson et al., 2019) is adopted to compute the similarity between the input dialog history x and the i-th candidate response c_i:

$$r_i = Cosine(A(x), A(c_i)) \in \mathbb{R}^1. \tag{3.4}$$

During training, we randomly sample 20 negative responses for each ground truth response. A cross-entropy loss is applied, aiming to rank the ground truth highest.

3.4 Methodology - ST

In this section, we introduce our self-training (ST) algorithm. The overall ST algorithm is introduced in Section 3.4.1, and a new text augmentation method (GradAug) for ST to train a more robust *Student* is elaborated in Section 3.4.2.

3.4.1 Overall ST Algorithm

During training, two data pools are maintained and denoted as U (unlabeled data) and L (labeled data). Two versions of the model are maintained, *Teacher* (F^T) and *Student* (F^S). Before the iterations of ST start, the *Teacher* is first trained on the initial small number of labeled data L to "warm-up".

Pseudo-Labeling. At the beginning of an ST iteration, the *Teacher* first makes predictions on U. For every data input $x \in U$, the *Teacher* predicts the label of x as $\hat{y}_x = F^T(x)$. We set the predicted score of the prediction $\hat{y}_x$ as the *confidence score* s_x for this prediction. When there is only a single label in the prediction $\hat{y}_x$ (c.f. intent

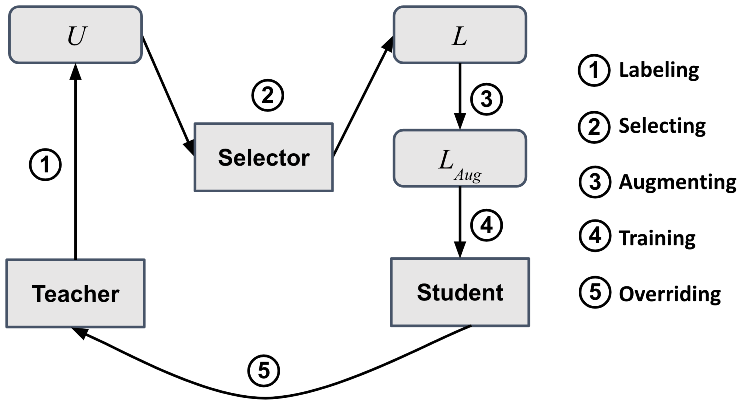


Figure 3.1 – Pipeline of one ST iteration. The *Teacher* first generates predictions for data in U. Then, the *Selector* chooses the most confident samples based on the *Teacher*'s predictions and assign pseudo labels to them before appending to L. Afterwards, L is augmented by "GradAug" to train a *Student*. Lastly, the trained *Student* becomes the *Teacher* in the next iteration. Multiple iterations are computed till the *Student* converges.

classification, response selection), s_x is the prediction score corresponding to the predicted label. When there are multiple labels in the prediction $\hat{y}_x$ (c.f. dialog state tracking, dialog act prediction), s_x takes the mean of the prediction scores corresponding to the predicted labels. In each iteration, the *Selector* chooses top-k instances from U with the highest confidence scores, and assigns the corresponding predictions $\hat{y}_x$ as labels to them. These labeled instances will be moved from U to L.

Iterative *Student* training. The updated L is used to train a stronger *Student* model. We applied dropout (Srivastava et al., 2014) and a new text augmentation technique (GradAug) introduced later in Section 3.4.2 which augments L to L_{Aug}. At the end of each iteration, the *Teacher* model is overridden by the current *Student* to be used in the next iteration. We reinitialize the *Student* in very iteration to avoid over-fitting the initial and earlier data in L in multiple training iterations. As noted by Xie et al. (2020); Du et al. (2020), the *Student* should have an equal or larger capacity than the *Teacher* to gradually learn from L with increasing size. In this chapter, we set the *Student* the same size as the *Teacher*, and we demonstrate in experiments that consistent improvements can be achieved without increasing model capacity.

Details of our ST algorithm are described in Algorithm 2, and the pipeline of one ST iteration (i.e., the "While" loop in Algorithm 2) is visualized in Figure 3.1.

3.4.2 Text Augmentation (GradAug)

Next, we propose a novel text augmentation technique called "GradAug" for data in L to train a more robust *Student*. Our method employs the masked language model

Algorithm 2 Self-training (ST) for ToD

```
Input: Labeled data: $L$, Unlabeled data: $U$, Teacher: $F^T$, Student: $F^S$, Number of
    pseudo-labeled data in an iteration: $k$, Number of augmentations per input: $q$
Output: A trained Student $F^S$
 1: Initialize $F^T$ and train $F^T$ on $L$
 2: while $F^S$ not good enough & $U \neq \emptyset$ do
 3:     Initialize $F^S$, $L' \leftarrow Priority\_list()$
 4:     for $x \in U$ do
 5:         Compute prediction label $\hat{y}_x = F^T(x)$
 6:         Compute confidence score $s_x$
 7:         $L'.insert(\{x, \hat{y}_x, s_x\})$
 8:     end for
 9:     $L' \leftarrow L'.top(k)$
10:     $L \leftarrow L \cup L'$, $U \leftarrow U \backslash L'$
11:     $L_{Aug} \leftarrow GradAug(L, F^T, q)$
12:     Train $F^S$ on $L_{Aug}$ with dropout
13:     $F^T \leftarrow F^S$
14: end while
```

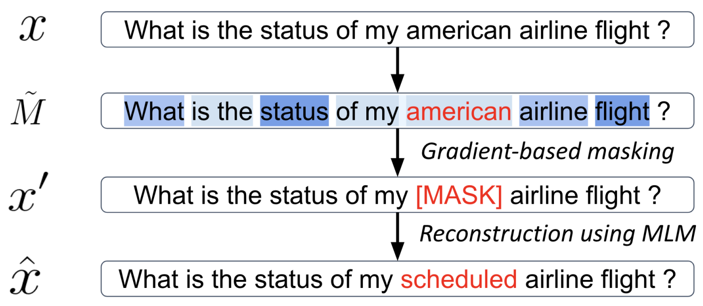
(MLM, Devlin et al. (2019a); Liu et al. (2019b)), which is a common pre-training strategy for BERT-like architectures. In MLM, some tokens are replaced by the special token [MASK], and the model is asked to reconstruct the original tokens from the context.

To utilize a pre-trained MLM (e.g. BERT) for text augmentation, the first step is to decide *which tokens to mask*. Random sampling is used by the original BERT framework and a recent text augmentation method (SSMBA, Ng et al. (2020)). However, if some crucial tokens are masked, the semantics might change after the reconstruction. For example, if the important token “status” in Figure 3.2 is masked, top predictions from the MLM of BERT includes “purpose”, “cost”, and “route”, which will potentially change the original semantics.

Gradient-based token masking. Instead of randomly masking tokens, we compute a *masking probability* $\boldsymbol{p} = [p_1, ..., p_n]$ for an input x of n tokens. For input x with token embedding matrix[1] $X = [X_1, ..., X_n]^\intercal \in \mathbb{R}^{n \times d}$ and label y, the *importance* of tokens in x to the label y is computed by a *saliency map* (Simonyan et al., 2014) $\boldsymbol{m}$:

$$\begin{cases} \boldsymbol{m} = \Big[M(X_1), \dots, M(X_n)\Big]^\intercal \in \mathbb{R}^n, \\ M(X_i) = \mathbb{1}^\intercal \left(\dfrac{\partial F_y^T(X)}{\partial X_i} \right) \in \mathbb{R}^1, \end{cases} \tag{3.5}$$

[1]We use the token embeddings of BERT-like architectures, rather than position or segmentation embeddings.



x
What is the status of my american airline flight ?
$\tilde{M}$
What is the status of my american airline flight ?
Gradient-based masking
x'
What is the status of my [MASK] airline flight ?
Reconstruction using MLM
$\hat{x}$
What is the status of my scheduled airline flight ?

Figure 3.2 – An illustrative example of GradAug. First, the smooth saliency $\tilde{M}$ is computed for each token, and we highlight important tokens in blue for the intent label "flight_status". Less important tokens are more likely to be masked. Then, the masked token ("american") is reconstructed by the MLM of BERT and the replacement token "scheduled" does not change the semantics of the original sentence.

where $F_y^T(X)$ is the *Teacher* model's prediction score for the label y. $M(X_i)$ measures the *importance* of the i-th token by accumulating the gradients of all elements in its embedding $X_i \in \mathbb{R}^d$ by differentiating $F_y^T(X)$ w.r.t. X_i. The intuition is that tokens with large gradients are important to the label y. However, previous studies (Sundararajan et al., 2017; Smilkov et al., 2017) pointed out that raw gradients can be very noisy and may sharply fluctuate locally. To this end, we compute a *smooth saliency* measure (Smilkov et al., 2017) $\tilde{M}(X_i)$ for the i-th token as:

$$\begin{cases} \tilde{M}(X_i) = \dfrac{1}{m}\displaystyle\sum_{j=1}^{m} M(\tilde{X}_i^j) \in \mathbb{R}^1, \\ \tilde{X}_i^j = X_i + \boldsymbol{z}^j, \end{cases} \tag{3.6}$$

where m Gaussian noises $\boldsymbol{z}^j \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \in \mathbb{R}^d$ with mean $\mathbf{0}$ and diagonal co-variance matrix $\mathbf{\Sigma}$ are added to X_i to calculate m regular saliency measures, which average to the *smooth saliency* $\tilde{M}(X_i)$ for X_i. The probability p_i of masking the i-th token is inversely correlated to $\tilde{M}(X_i)$ as:

$$p_i \propto \frac{1}{\tilde{M}(X_i)^\beta}, \tag{3.7}$$

where β controls the flatness of the distribution $\boldsymbol{p}$, and $\boldsymbol{p}$ is normalized by its sum. As the probability p_i to mask a token x_i is inversely correlated to its importance $\tilde{M}(X_i)$ to a downstream task, more important tokens are less likely to be masked. We sample 15% [2] tokens of x based on $\boldsymbol{p}$ and replace them by [MASK] to corrupt x to x'. As F^T is updated in each ST iteration, $\boldsymbol{p}$ is dynamically calculated in each ST iteration.

Reconstruction using MLM. To reconstruct the masked tokens in x', we utilize a pre-trained MLM to predict the [MASK] tokens. For stochastic purposes suggested by

[2]This is the default ratio used by BERT and SSMBA.

Algorithm 3 GradAug

```
Input: Labeled data: $L$, Teacher: $F^T$, Number of augmentations per input: $q$
Output: Augmented labeled data $L_{Aug}$
1: Initialize $L_{Aug} \leftarrow L$
2: for $\{x, y\} \in L$ do
3:     Compute masking probability $\boldsymbol{p}$ using $F^T$
4:     for $j \in 1 \ldots q$ do
5:         $x' \leftarrow$ Mask tokens of $x$ based on $\boldsymbol{p}$
6:         $\hat{x} \leftarrow$ Predict masked tokens by MLM
7:         $L_{Aug}.append(\{\hat{x}, y\})$
8:     end for
9: end for
```

Fan et al. (2018), we reconstruct each [MASK] by sampling one token from the ten most likely tokens according to their predicted probabilities. Afterward, we get a paraphrased $\hat{x}$ of the original x as an augmentation. As our gradient-based masking scheme avoids replacing tokens crucial to the meaning of x, the label of $\hat{x}$ is preserved the same as x.

An illustrative example of GradAug is given in Figure 3.2, and the detailed procedure applying GradAug on L is described in Algorithm 3.

3.5 Evaluation

3.5.1 Datasets and Experiment Settings

We evaluate four different datasets for four downstream tasks as in Wu et al. (2020).

OOS (Larson et al., 2019) is a benchmark dataset for intent classification in ToD. It consists of 150 in-domain intents and 1 out-of-scope intent. The full dataset contains 15,100/3,100/5,500 samples for train/validation/test, and all data are balanced across 151 different intents.

MWOZ (Eric et al., 2020) is evaluated in three downstream tasks, including dialog state tracking, dialogue act prediction, and response prediction. It contains 8,420/1,000/1,000 dialogues for train/validation/test. For dialog act prediction, we remove the domain information from original labels as in Wu et al. (2020), resulting 13 DA intents.

DSTC2 (Henderson et al., 2014) and **GSIM** (Shah et al., 2018) are two corpus used in dialog act prediction and response selection tasks. DSTC2 contains 1,612/506/1,117 dialogues for train/validation/test; GSIM contains 1,500/469/1,039 dialogues for train/-validation/test. DA intent labels of DSTC2 and GSIM are mapped to universal dialogue acts (Paul et al., 2019), resulting in 19 and 13 DA intents respectively.

To construct different few-shot learning scenarios with unlabeled data, we randomly sample 1% or 10% of the training data to serve as the initial labeled data L, while the remaining are used as unlabeled data U. We report results averaged over three different random seeds for each experiment to reduce data sampling variance. We also report the upper bound of pre-trained models *without* ST using all labeled training data, referred to as "Full".

We test two pre-trained models: (i). uncased base BERT with 110M parameters; (ii). ToD-BERT [3] (Wu et al., 2020) that is further pre-trained on nine public ToD datasets on top of BERT. When ST is applied to them, the corresponding MLM is used by GradAug to reconstruct masked tokens. Basic model parameters of the first three downstream tasks are set the same as Wu et al. (2020). In response selection, we reduced the training batch size from 25 to 20 to fit our computation constraint. BERT and Tod-BERT without ST are trained on the initial labeled data until validation performance does not improve for 20 epochs [4]. When the *Student* is trained on L_{Aug} (c.f. Algorithm 2 line 12), we apply early stop until validation performance does not improve for 10 epochs. If *Student's* validation performance does not improve for 3 ST iterations (c.f. Algorithm 2 line 2), we stop ST. The number (k) of pseudo-labeled data in each ST iteration is set as the initial size of L; the number (q) of augmentations per input for GradAug is set to 3; β in Equation 3.7 is set to 1. An exhaustive search on hyper-parameters (k, q, β) is not conducted because it is expensive to conduct for large pre-trained models on all four downstream tasks. We expect better results of ST can be achieved with a thorough hyper-parameter search by researchers without computation constraints.

3.5.2 Main Results of Four Downstream Tasks

Intent classification. Results of intent classification on OOS are presented in Table 3.1 with *accuracy* of **all** 151 intents; 150 **in**-domain intents; the **out**-of-scope intent, and the *recall* of the out-of-scope intent. ST significantly improves the pre-trained BERT and ToD-BERT. When only 1% labeled data are used, ST achieves 33.6% and 36.8% higher accuracy on all 151 intents for BERT and ToD-BERT respectively. For 10% labeled data, the above two margins are 7.0% and 9.8%. Furthermore, ST largely improves the recall of the out-of-scope intent, which means that it is robust to the out-of-scope intent with more noisy distributions.

Dialog state tracking. Results of dialog state tracking on MWOZ are presented in Table 3.2. Two common evaluation metrics (Budzianowski et al., 2018; Wu et al., 2019a) are used: *slot accuracy* and *joint goal accuracy.* Slot accuracy is computed for each state (domain, slot, value) to check whether the value is correctly predicted. Joint goal

[3]We used their joint version (ToD-BERT-jnt) pre-trained with the MLM and "response contrastive loss" objectives

[4]Our different (often better) results compared to the ToD-BERT paper mainly come from this stricter early stop criteria.

Data	Model	Acc. (all)	Acc. (in)	Acc. (out)	Recall (out)
1%	BERT	36.5%	44.6%	81.8%	0.2%
	BERT-ST	**70.1%**	**82.2%**	**84.3%**	**15.5%**
	ToD-BERT	39.0%	47.1%	82.0%	2.3%
	ToD-BERT-ST	**75.8 %**	**87.8%**	**85.5%**	**21.9%**
10%	BERT	73.6 %	87.4%	83.9%	11.7%
	BERT-ST	**80.6%**	**94.3%**	**84.9%**	**17.1%**
	ToD-BERT	75.5%	89.4%	84.1%	13.3%
	ToD-BERT-ST	**85.3%**	**94.7%**	**89.4%**	**42.8%**
Full*	BERT	84.9%	95.8%	88.1%	35.6%
	ToD-BERT	86.6%	96.2%	89.9%	43.6%

Table 3.1 – Results of intent classification. Bold numbers indicate ST improves the corresponding pre-trained model. Results with * are taken from Wu et al. (2020).

Data	Model	Joint Acc	Slot Acc
1%	BERT	8.0%	84.3%
	BERT-ST	**8.8%**	**84.5%**
	ToD-BERT	8.4%	85.7%
	ToD-BERT-ST	**9.9%**	**86.5%**
10%	BERT	21.2%	92.0%
	BERT-ST	**23.9%**	**92.4%**
	ToD-BERT	25.5%	93.4%
	ToD-BERT-ST	**28.3%**	**93.7%**
Full*	BERT	45.6%	96.6%
	ToD-BERT	48.0%	96.9%

Table 3.2 – Results of dialog state tracking. Bold numbers indicate ST improves the corresponding pre-trained model. Results with * are taken from Wu et al. (2020).

accuracy checks whether the predicted states exactly matches the ground truth states. We can see that ST consistently improves both BERT and ToD-BERT. E.g., ST has 1.5% and 2.8% joint goal accuracy improvement over ToD-BERT when 1% and 10% labeled data are used respectively. Similar margins can be observed for ST on top of BERT.

Dialog act prediction. Experiments are conducted on three datasets, and results are reported in Table 3.3. We report *micro-F1* and *macro-F1* scores for this multi-label classification task. Again, the benefit of ST can be observed by the improvement for both BERT and ToD-BERT. When 10% labeled data are used, BERT and ToD-BERT perform similarly to their upper bound (Full), and the improvement margin of ST is limited. When 1% labeled data are used, more notable margins of ST can be seen on the two simpler datasets (DSTC2, GSIM) and the macro-F1 score of MWOZ.

Response selection. Results of response selection on three datasets are reported

Data	Model	MWOZ		DSTC2		GSIM	
		micro-F1	macro-F1	micro-F1	macro-F1	micro-F1	macro-F1
1%	BERT	83.5%	61.2%	79.1%	26.8%	70.3%	27.9%
	BERT-ST	82.7%	**64.2%**	**81.4%**	**27.3%**	**73.0%**	**29.8%**
	ToD-BERT	85.8%	67.0%	80.9%	25.3%	87.5%	37.6%
	ToD-BERT-ST	**86.9%**	**71.8%**	**82.7%**	**28.5%**	**92.6%**	**40.8%**
10%	BERT	89.8%	77.8%	88.9%	35.7%	97.1%	44.1%
	BERT-ST	89.5%	**79.2%**	**92.3%**	**38.4%**	**97.6%**	**44.6%**
	ToD-BERT	90.0%	78.4%	90.6%	38.8%	98.6%	44.9%
	ToD-BERT-ST	**90.2%**	**79.6%**	**92.9%**	**40.5%**	**99.3%**	**45.6%**
Full*	BERT	91.4%	79.7%	92.3%	40.1%	98.7%	45.2%
	ToD-BERT	91.7%	80.6%	93.8%	41.3%	99.5%	45.8%

Table 3.3 – Results of dialog act prediction. Bold numbers indicate ST improves the corresponding pre-trained model. Results with * are taken from Wu et al. (2020).

Data	Model	MWOZ		DSTC2		GSIM	
		Recall@1	Recall@3	Recall@1	Recall@3	Recall@1	Recall@3
1%	BERT	7.3%	19.5%	3.8%	9.8%	4.0%	11.4%
	BERT-ST	**23.8%**	**46.1%**	**36.7%**	**51.1%**	**11.1%**	**24.2%**
	ToD-BERT	37.5%	63.0%	35.7%	53.8%	11.4%	24.1%
	ToD-BERT-ST	**43.5%**	**66.3%**	**48.0%**	**64.6%**	**27.8%**	**42.9%**
10%	BERT	26.1%	56.5%	27.7%	42.9%	13.4%	28.3%
	BERT-ST	**43.1%**	**66.1%**	**53.7%**	**67.1%**	**22.3%**	**40.4%**
	ToD-BERT	47.2%	69.4%	51.3%	66.0%	28.5%	47.8%
	ToD-BERT-ST	**60.2%**	**81.9%**	**58.8%**	**72.2%**	**41.8%**	**64.9%**
Full	BERT	47.5%	75.5%	46.6%	62.1%	13.4%	32.9%
	ToD-BERT	66.9%	89.1%	59.5%	73.1%	43.0%	65.3%

Table 3.4 – Results of response selection. Bold numbers indicate ST improves the corresponding pre-trained model.

in Table 3.4. We randomly sample 100 responses as negative responses and report Recall@1&3 (Henderson et al., 2019) indicating whether the true response is ranked in the top-1 or top-3 predicted responses. ST improves the two pre-trained models by large margins. When 1% labeled data are used, ST achieves 6%, 12.3%, and 16.4% higher Recall@1 accuracy over ToD-BERT on three datasets respectively. For 10% labeled data, the three margins above are 13.0%, 7.5%, and 14.4% respectively. Larger improvements can be observed for ST on top of BERT.

Altogether, our experiments on four different downstream tasks reveal that:

- Self-training provides complementary benefits on top of pre-training. ST consistently improves both BERT and ToD-BERT on all four downstream tasks with only 1% and 10% labeled data.
- Self-training is on par with customized pre-training for ToD. BERT performs worse than ToD-BERT, yet BERT-ST achieves comparable or even better performance

	IC Acc. (all)	RS Recall@3
ToD-BERT-ST	85.3%	64.9%
w/o Smooth Saliency	81.9%	64.4%
w/o Augmentation	80.4%	54.8%
w/o Pseudo-Labeling	76.9%	49.7%
ToD-BERT	75.5%	47.8%

Table 3.5 – Ablation study of ST for intent classification (IC) on OOS and response selection (RS) on GSIM.

than ToD-BERT that is heavily pre-trained on ToD corpora.

- Self-training bridges the gap between few-shot learning and full supervision. BERT and ToD-BERT with 10% labeled data perform much worse than models using all labeled data (“Full”) for intent classification and response selection. ST largely improves performances in these two cases with results comparable to “Full”.

- The benefit of self-training is evident on two simpler single-label prediction tasks (intent classification, response selection), indicated by 6-37% gain with 1% labeled data; 7-15% gain with 10% labeled data. The margin is smaller on two other more challenging multi-label prediction tasks (dialog state tracking, dialog act prediction) that also require more complex reasoning over dialog history.

3.5.3 In-depth Analysis

In this section, we provide several in-depth analyses of the proposed self-training approach. As case studies, we limited our discussion on intent classification (**IC**) on OOS and response selection (**RS**) on GSIM using ToD-BERT-ST with 10% labeled data. Reported results are accuracies on all intents and Recall@3 respectively.

Ablation study. In Table 3.5, we compare three simplified versions of ToD-BERT-ST to understand the effects of different components. We can observe that: (i) Masking tokens using the smooth saliency computed in Eq. (3.6) for GradAug is beneficial because replacing it by the vanilla saliency in Eq. (3.5) (“w/o Smooth Saliency”) degrades the performance by 3.4% and 0.5% on IC and RS. (ii) Training a more robust *Student* using data augmented by GradAug is advantageous because dropping this augmentation step (“w/o Augmentation”) impairs performance by 4.9% and 10.1%. (iii) The Pseudo-Labeling operation to iteratively label unlabeled data is important for ST, indicated by the 8.4% and 15.2% performance drop of “w/o Pseudo-Labeling” that only applies GradAug to the initial labeled data without utilizing unlabeled data.

	IC **Acc. (all)**	**RS** **Recall@3**
Top-k (Ours)	85.3%	64.9%
Random-k	84.0%	64.1%
Least-k	82.7%	61.4%
Select-all	76.0%	50.8%

Table 3.6 – Comparison to other *Selectors* in ST for intent classification (IC) on OOS and response selection (RS) on GSIM.

	IC **Acc. (all)**	**RS** **Recall@3**
GradAug (Ours)	85.3%	64.9%
SSMBA (Ng et al., 2020)	84.6%	64.2%
MixText (Chen et al., 2020a)	83.6%	62.7%
EDA (Wei and Zou, 2019)	77.2%	57.6%
w/o Augmentation	80.4%	54.8%

Table 3.7 – Comparison to other text augmentation methods to train the *Student* for intent classification (IC) on OOS and response selection (RS) on GSIM.

Comparison to other *Selectors* in ST. In Table 3.6, we compare our scheme of selecting samples with top-k confident predictions from U in each iteration with (i) Random-k: randomly select k samples; (ii) Least-k: select samples with least-k confident predictions (iii) Select-all (Xie et al., 2020; Du et al., 2020): label all samples of U in an iteration and relabel them in the next iteration. We could see that "Random-k" and "Least-k" perform worse than ours, yet they both outperform "Select-all" by large margins. We believe that it is because the initial *Teacher* trained on limited labeled data is not good enough to assign reliable labels to a large number of unlabeled data. Explorations on other sophisticated sample selection schemes are orthogonal to the focus of this chapter and will be left as future work.

Comparison to other text augmentation methods. In Table 3.7, we compare GradAug with three representative text augmentation methods to augment L for training the *Student*. We follow the default setting of these techniques and apply them to our ST pipeline to generate three paraphrases for each input as in GradAug. GradAug performs better than the current state-of-the-art (SSMBA, MixText) and outperforms EDA by large margins. As EDA might easily change the input semantics, it even performs worse than using no data augmentation for intent classification. This result reinforces the importance of preserving semantics during augmentation for ToD.

3.6 Chapter Summary

We study using self-training to improve the strong pre-trained models for few-shot learning tasks in ToD. An iterative self-training method with a new text augmentation technique (GradAug) is proposed to gradually train a stronger *Student* model using unlabeled data. Extensive empirical results on four different downstream tasks (intent classification, dialog state tracking, dialog act prediction, and response selection) in ToD demonstrate the consistent improvements of self-training on top of state-of-the-art pre-trained models. Our in-depth analysis demonstrates that the proposed sample labeling scheme and the new text augmentation technique are both critical for the final performance. Our findings on using self-training to improve learning from limited labeled data may inspire future studies to build scalable ToD systems or other application scenarios with rich unlabeled data. For chapters in the next part, we study the knowledge retention challenge to mitigate the catastrophic forgetting issue of neural networks in different applications.

Knowledge Retention Part II

4 Continual Learning with Knowledge Retention in ToD Systems

4.1 Introduction

Existing NLG models (Wen et al., 2015b; Tran and Nguyen, 2017; Tseng et al., 2018) in ToD system are typically trained offline using annotated data from a single or a fixed set of domains. However, a scalable ToD system in real-life applications often needs to expand its knowledge to new domains and functionalities. Therefore, it is crucial to develop an NLG approach with the capability of continual learning after a dialog system is deployed. Recently, Mi et al. (2019); Qian and Yu (2019); Peng et al. (2020b) studied learning new domains with limited training data as in our Chapter 2. However, existing methods only consider a one-time adaptation process. The continual learning setting and the corresponding knowledge retention issue remain to be explored. Specifically, an NLG model should be able to continually learn new utterance patterns without forgetting the old ones it has already learned.

We diagnose in Section 4.4.3 that neural NLG models suffer the detrimental catastrophic forgetting issue when continually trained on new domains. A naive solution is to retrain the NLG model using all historical data every time. However, it is not scalable due to severe computation and storage overhead. To this end, we store a small number of representative utterances from previous data, namely *exemplars*, and replay them to the NLG model each time when it needs to be trained on new data. In this chapter, we propose a *prioritized* exemplar selection scheme to choose representative and diverse exemplar utterances for NLG. We empirically demonstrate that the prioritized exemplar replay helps to alleviate catastrophic forgetting by a large degree.

In practice, the number of exemplars should be reasonably small to maintain a manageable memory footprint. Therefore, the constraint of not forgetting old utterance patterns is not strong enough. To enforce a stronger constraint, we propose a regularization method

This chapter is based on the paper (Mi et al., 2020a) published in the conference on Findings of Empirical Methods in Natural Language Processing (Findings of EMNLP, 2020).

based on the well-known technique, Elastic Weight Consolidation (EWC (Kirkpatrick et al., 2017)). The idea is to use a quadratic term to regularize the parameters that are important for previous data elastically. Besides the wide application in computer vision, EWC has been recently applied to the domain adaptation task for Neural Machine Translation (Thompson et al., 2019; Saunders et al., 2019). In this chapter, we combine EWC with exemplar replay by approximating the Fisher Information Matrix w.r.t. the carefully chosen exemplars so that not all historical data need to be stored. Furthermore, we propose to adaptively adjust the regularization weight to consider the difference between new and old data to flexibly deal with different new data distributions.

To summarize our contribution: (1) to the best of our knowledge, this is the first attempt to study the practical continual learning configuration for NLG in task-oriented dialog systems; (2) we propose a method called Adaptively Regularized Prioritized Exemplar Replay (*ARPER*) for this task and benchmark it with a wide range of state-of-the-art continual learning techniques; (3) extensive experiments are conducted on the MultiWOZ (Budzianowski et al., 2018) dataset to continually learn new tasks, including domains and DA intents using three base NLG models. Empirical results demonstrate the superior performance of *ARPER* and its ability to mitigate catastrophic forgetting. Our code is available at https://github.com/MiFei/Continual-Learning-for-NLG

4.2 Related Work

The background of neural NLG models is reviewed in Section 2.2. In this section, we review some works studying continual learning and the catastrophic forgetting issue in other NLP tasks.

The major challenge for continual learning is catastrophic forgetting (McCloskey and Cohen, 1989; French and Chater, 2002). Methods designed to mitigate catastrophic forgetting fall into three categories: regularization (Li and Hoiem, 2018; Kirkpatrick et al., 2017; Zenke et al., 2017a), exemplar replay (Rebuffi et al., 2017; Chaudhry et al., 2019; Castro et al., 2018) and dynamic architectures (Rusu et al., 2016; Maltoni and Lomonaco, 2019). Methods using dynamic architectures increase model parameters throughout the training process, which leads to an unfair comparison with other methods. In this work, we focus on the first two categories.

Regularization methods add specific regularization terms to consolidate knowledge learned before. Li and Hoiem (2018) introduces knowledge distillation (Hinton et al., 2015) to penalize model logit change, and it is widely employed by Rebuffi et al. (2017); Castro et al. (2018); Wu et al. (2019c); Hou et al. (2019); Zhao et al. (2020). Kirkpatrick et al. (2017); Zenke et al. (2017a); Aljundi et al. (2018) propose to penalize changes on parameters that are crucial to old knowledge according to various importance measures.

Exemplar replay methods store past samples, a.k.a *exemplars*, and replay them periodically to prevent the model from forgetting previous knowledge. Besides selecting exemplars uniformly, Rebuffi et al. (2017) incorporates the *Herding* technique (Welling, 2009) to select exemplars, and it soon becomes popular (Castro et al., 2018; Wu et al., 2019c; Hou et al., 2019; Zhao et al., 2020; Mi et al., 2020b). Ramalho and Garnelo (2019) proposes to store the most "surprising" samples that the model is least confident. Chaudhry et al. (2019) demonstrated the effectiveness of exemplars for various continual learning tasks in computer vision. Instead of storing raw samples, Shin et al. (2017); Riemer et al. (2019) use generative models, such as Generative Adversarial Network (Goodfellow et al., 2014) or Variational AutoEncoder (Kingma and Welling, 2014), to generate virtual samples akin to past data.

The catastrophic forgetting issue in NLP tasks has raised increasing attention recently (Mou et al., 2016; Chronopoulou et al., 2019). Yogatama et al. (2019); Arora et al. (2019) identified the detrimental catastrophic forgetting issue while fine-tuning ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019b). To deal with this issue, He et al. (2019a) proposed to replay pre-train data during fine-tuning heavily, and Chen et al. (2020b) proposed an improved Adam optimizer to recall knowledge captured during pre-training. The catastrophic forgetting issue is also noticed in domain adaptation setups for neural machine translation (Saunders et al., 2019; Thompson et al., 2019; Varis and Bojar, 2019) and the reading comprehension task (Xu et al., 2020). (Saunders and Byrne, 2020) later applied EWC (Kirkpatrick et al., 2017) to retain the gender balance of translated sentences during fine-tuning.

Lee (2017) firstly studied the continual learning setting for dialog state tracking in task-oriented dialog systems. However, their setting is still a one-time adaptation process, and the adopted dataset is small. Shen et al. (2019) recently applied progressive network (Rusu et al., 2016) for the semantic slot filling task from a continual learning perspective similar to ours. However, their method is based on a dynamic architecture that is beyond the scope of this chapter. Liu et al. (2019a) proposed a Boolean operation of "conceptor" matrices for continually learning sentence representations using linear encoders. Li et al. (2020) combined continual learning and language systematic compositionality for sequence-to-sequence learning tasks.

4.3 Methodology - ***ARPER***

In this section, we first introduce the background of neural NLG models in Section 4.3.1, and the continual learning formulation in Section 4.3.2. In Section 4.3.3, we introduce the proposed method *ARPER*.

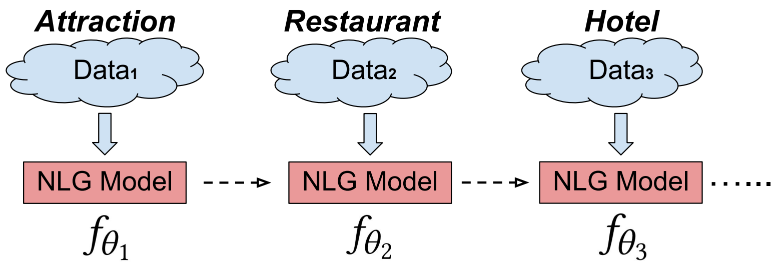


Figure 4.1 – An example for a NLG model to continually learn new domains. The model needs to perform well on all domains it has seen before. For example f_{θ_3} needs to deal with all three previous domains (*Attraction, Restaurant, Hotel*).

4.3.1 Background on Neural NLG Models [1]

The NLG component of task-oriented dialog systems is to produce natural language utterances conditioned on a *semantic representation* called dialog act (DA). Specifically, the dialog act $\mathbf{d}$ is defined as the combination of *intent* $\mathbf{I}$ and a set of slot-value pairs $S(\mathbf{d}) = \{(s_i, v_i)\}_{i=1}^{p}$:

$$\mathbf{d} = [\underbrace{\mathbf{I}}_{\text{Intent}}, \underbrace{(s_1, v_1), \ldots, (s_p, v_p)}_{\text{Slot-value pairs}}], \tag{4.1}$$

where p is the number of slot-value pairs. Intent $\mathbf{I}$ controls the utterance functionality, while slot-value pairs contain information to express. For example, "*There is a restaurant called [La Margherita] that serves [Italian] food.*" is an utterance corresponding to a DA "*[Inform, (name=La Margherita, food=Italian)]*"

Neural models have recently shown promising results for NLG tasks. Conditioned on a DA, a neural NLG model generates an utterance containing the desired information word by word. For a DA $\mathbf{d}$ with the corresponding ground truth utterance $\mathbf{Y} = (y_1, y_1, ..., y_K)$, the probability of generating $\mathbf{Y}$ is factorized as below:

$$f_\theta(\mathbf{Y}, \mathbf{d}) = \prod_{k=1}^{K} p_{y_k} = \prod_{k=1}^{K} p(y_k | y_{<k}, \mathbf{d}, \theta), \tag{4.2}$$

where f_θ is the NLG model parameterized by θ, and p_{y_k} is the output probability (i.e. softmax of logits) of the ground truth token y_k at position k. The typical objective function for an utterance $\mathbf{Y}$ with DA $\mathbf{d}$ is the average cross-entropy loss w.r.t. all tokens in the utterance (Wen et al., 2015b,c; Tran and Nguyen, 2017; Peng et al., 2020b):

$$\mathcal{L}_{CE}(\mathbf{Y}, \mathbf{d}, f_\theta) = -\frac{1}{K} \sum_{k=1}^{K} log(p_{y_k}) \tag{4.3}$$

[1]This part recaps some contents in Section 2.2 for context coherence.

4.3.2 Continual Learning of NLG

In practice, an NLG model needs to continually learn new domains or functionalities. Without loss of generality, we assume that new data arrive *phase by phase* (Rebuffi et al., 2017; Kirkpatrick et al., 2017). In a new phase t, new data $\mathbf{D}_t$ are used to train the NLG model $f_{\theta_{t-1}}$ obtained till the last phase. The updated model f_{θ_t} needs to perform well on *all* phases so far. A phase can be defined with different modalities to reflect diverse real-life applications. In subsequent experiments, we consider continually learning new domains and DA intents. An example setting of continually learning new domains is illustrated in Figure 4.1.

We emphasize that the setting of continual learning is different from that of domain adaptation. The latter is a one-time adaptation process, and the focus is to optimize performance on a target domain transferred from source domains but without considering potential performance drop on source domains (Mi et al., 2019; Qian and Yu, 2019; Peng et al., 2020b). In contrast, continual learning requires an NLG model to continually learn new phases in multiple transfers, and the goal is to make the model perform well on all phases learned so far.

4.3.3 Adaptively Regularized Prioritized Exemplar Replay (***ARPER***)

In this section, we introduce the proposed method (*ARPER*) with prioritized exemplar replay and an adaptive regularization technique to further alleviate the catastrophic forgetting issue.

Prioritized Exemplar Replay

To prevent the NLG model catastrophically forgetting utterance patterns in earlier phases, a small subset of a phase's utterances are selected as *exemplars*, and exemplars in previous phases are *replayed* to the later phases. During training the NLG model f_{θ_t} for phase t, the set of exemplars in previous phases, denoted as $\mathbf{E}_{1:t-1} = \{\mathbf{E}_1, \ldots, \mathbf{E}_{t-1}\}$, is replayed by joining with the data $\mathbf{D}_t$ of the current phase. Therefore, the training objective with exemplar replay can be written as:

$$\mathcal{L}_{ER}(\theta_t) = \sum_{\{\mathbf{Y},\mathbf{d}\} \in \mathbf{D}_t \cup \mathbf{E}_{1:t-1}} \mathcal{L}_{CE}(\mathbf{Y}, \mathbf{d}, f_{\theta_t}). \tag{4.4}$$

The set of exemplars of phase t, referred to as $\mathbf{E}_t$, is selected after f_{θ_t} has been trained and will be replayed to later phases.

The quality of exemplars is crucial to preserve the performance on previous phases. We propose a *prioritized* exemplar selection method to select representative and diverse

Algorithm 4 *ARPER.SELECT_EXEMPLARS*: Prioritized exemplar selection procedure for phase t

```
1: procedure SELECT_EXEMPLARS($\mathbf{D}_t, f_{\theta_t}, m$)
2:     $\mathbf{E}_t \leftarrow$ new $Priority\_list()$
3:     $\mathbf{D}_t \leftarrow sort(\mathbf{D}_t, key = U, order = asc)$
4:     while $|\mathbf{E}_t| < m$ do
5:         $\mathbf{S}_{seen} \leftarrow$ new $Set()$
6:         for $\{\mathbf{Y}, \mathbf{d}\} \in \mathbf{D}_t$ do
7:             if $S(\mathbf{d}) \in \mathbf{S}_{seen}$ then continue
8:             else
9:                 $\mathbf{D}_t.remove(\{\mathbf{Y}, \mathbf{d}\})$
10:                $\mathbf{E}_t.insert(\{\mathbf{Y}, \mathbf{d}\})$
11:                $\mathbf{S}_{seen}.insert(S(\mathbf{d}))$
12:                if $|\mathbf{E}_t| == m$ then
13:                    return $\mathbf{E}_t$
14:                end if
15:            end if
16:        end for
17:    end while
18: end procedure
```

utterances as follows.

Representative utterances. The first criterion is that exemplars $\mathbf{E}_t$ of a phase t should be representative of $\mathbf{D}_t$. We propose to select $\mathbf{E}_t$ as a *priority list* from $\mathbf{D}_t$ that *minimize* a priority score:

$$U(\mathbf{Y}, \mathbf{d}) = \mathcal{L}_{CE}(\mathbf{Y}, \mathbf{d}, f_{\theta_t}) \cdot |S(\mathbf{d})|^{\beta}, \tag{4.5}$$

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 $\mathcal{L}_{CE}$. Intuitively, the NLG model f_{θ_t} trained on $\mathbf{D}_t$ should be confident with representative utterances of $\mathbf{D}_t$, i.e., low $\mathcal{L}_{CE}$. However, $\mathcal{L}_{CE}$ is agnostic to the number of slots. We found that an utterance with many common slots in a phase could also have very low $\mathcal{L}_{CE}$, yet using such utterances as exemplars may lead to overfitting and thus forgetting of previous general knowledge. The second term $|S(\mathbf{d})|^{\beta}$ controls the importance of the number of slots in an utterance to be prioritized as exemplars. We empirically found in experiments that the best β is 0.5 (larger than 0).

Diverse utterances. The second criterion is that exemplars should contain diverse slots of the phase, rather than being similar or repetitive. A drawback of the above priority score is that similar or duplicated utterances containing the same set of frequent slots could be prioritized over utterances w.r.t. a diverse set of slots. To encourage diversity of selected exemplars, we propose an iterative approach to add data from $\mathbf{D}_t$ to the priority list $\mathbf{E}_t$ based on the above priority score. At each iteration, if the set of slots of the current utterance is already covered by utterances in $\mathbf{E}_t$, we skip it and move on to the data with the next best priority score.

Algorithm 1 shows the procedure to select m exemplars as a priority list $\mathbf{E}_t$ from $\mathbf{D}_t$. The outer loop allows multiple passes through $\mathbf{D}_t$ to select various utterances for the same set of slots $S(\mathbf{d})$.

Reducing Exemplars in Previous Phases

Algorithm 1 requires the number of exemplars to be given. A straightforward choice is to store the same and fixed number of exemplars for each phase as in Castro et al. (2018); Wu et al. (2019c); Hou et al. (2019). However, there are two drawbacks in this method: (1). the memory usage increases linearly with the number of phases; (2) it does not discriminate phases with different difficulty levels.

To this end, we propose to store 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 phases are continually learned, exemplars in previous phases are gradually reduced by only keeping the ones in the *front* of the priority list[2]. The exemplar size of a phase is set to be proportional to the training data size of the phase to differentiate the phase's difficulty. To be specific, suppose M exemplars are kept in total. The number of exemplars for a phase is:

$$|\mathbf{E}_i| = M \cdot \frac{|\mathbf{D}_i|}{\sum_{j=1}^{t} |\mathbf{D}_j|}, \forall i \in 1, \ldots, t, \tag{4.6}$$

where we choose 250/500 for M in experiments.

Constraint with Adaptive Elastic Weight Consolidation

Although exemplars of previous phases are stored and replayed, the size of exemplars should be reasonably small ($M \ll |\mathbf{D}_{1:t}|$) to reduce memory overhead. As a consequence, the constraint we have made to prevent the NLG model from catastrophically forgetting previous utterance patterns is not strong enough. To enforce a stronger constraint, we propose a regularization method based on the well-known Elastic Weight Consolidation (EWC, Kirkpatrick et al., 2017) technique.

[2]the priority list implementation allows reducing exemplars in constant time for each phase

Elastic weight consolidation (EWC). EWC utilizes a quadratic term to elastically regularize parameters important for previous phases. The loss function of using the EWC regularization together with exemplar replay for phase t can be written as:

$$\mathcal{L}_{ER\_EWC}(\theta_t) = \mathcal{L}_{ER}(\theta_t) + \lambda \sum_{i}^{N} F_i(\theta_{t,i} - \theta_{t-1,i})^2 \tag{4.7}$$

where N is the number of model parameters; $\theta_{t-1,i}$ is the i-th converged parameter of the model trained till the previous phase; $F_i = \nabla^2 \mathcal{L}_{CE}^{\mathbf{E}_{1:t-1}}(\theta_{t-1,i})$ is the i-th diagonal element of the Fisher Information Matrix approximated w.r.t. the set of previous exemplars $\mathbf{E}_{1:t-1}$. F_i measures the importance of $\theta_{t-1,i}$ to previous phases represented by $\mathbf{E}_{1:t-1}$. Typical usages of EWC compute F_i w.r.t. a uniformly sampled subset from historical data. In contrast, we propose to compute F_i w.r.t. the carefully chosen $\mathbf{E}_{1:t-1}$ so that not all historical data need to be stored. The scalar λ controls the contribution of the quadratic regularization term. The idea is to elastically penalize changes on parameters important (with large F_i) to previous phases, and more plasticity is assigned to parameters with small F_i.

Adaptive regularization. In practice, new phases have different difficulties and similarities compared to previous phases. Therefore, the degree of need to preserve the previous knowledge varies. To this end, we propose an adaptive weight (λ) for the EWC regularization term as follows:

$$\lambda = \lambda_{base} \sqrt{V_{1:t-1}/V_t}, \tag{4.8}$$

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

Algorithm 2 summarizes the continual learning procedure of *ARPER* for phase t. θ_t is initialized with θ_{t-1}, and it is trained with prioritized exemplar replay and adaptive EWC in Eq. (4.7). After training θ_t, exemplars $\mathbf{E}_t$ of phase t are computed by Algorithm 1, and exemplars in previous phases are reduced by keeping the most prioritized ones to preserve the total exemplar size.

Algorithm 5 *ARPER.LEARN_PHASE*: Procedure of *ARPER* to learn phase t

```
1: procedure LEARN_PHASE($\mathbf{D}_t, \mathbf{E}_{1:t-1}, f_{\theta_{t-1}}, M$)
2:     $\theta_t \leftarrow \theta_{t-1}$
3:     while $\theta_t$ not converged do
4:         $\theta_t \leftarrow update(\mathcal{L}_{ER\_EWC}(\theta_t))$
5:     end while
6:     $m \leftarrow M \cdot \frac{|\mathbf{D}_t|}{\Sigma_{j=1}^{t}|\mathbf{D}_j|}$
7:     $\mathbf{E}_t \leftarrow select\_exemplars(\mathbf{D}_t, f_{\theta_t}, m)$
8:     for $j = 1$ to $t-1$ do
9:         $\mathbf{E}_j \leftarrow \mathbf{E}_j.top(M \cdot \frac{|\mathbf{D}_j|}{\Sigma_{j=1}^{t}|\mathbf{D}_j|})$
10:    end for
11:    return $f_{\theta_t}, \mathbf{E}_t$
12: end procedure
```

4.4 Evaluation

4.4.1 Dataset and Evluation Metric

We use the MultiWOZ dataset [3] (Budzianowski et al., 2018) containing six domains (*Attraction, Hotel, Restaurant, Booking, Taxi and Train*) and seven DA intents (*"Inform, Request, Select, Recommend, Book, Offer-Booked, No-Offer"*). The original train/validation/test splits are used. For methods using exemplars, both training and validation set are continually expanded with exemplars extracted from previous phases.

To support experiments on continual learning new domains, we pre-process the original dataset by segmenting multi-domain utterances into single-domain ones. For instance, an utterance *"The ADC Theatre is located on Park Street. Before I find your train, could you tell me where you would like to go?"* is split into two utterances with domain "*Attraction*" and "*Train*" separately. If multiple sentences of the same domain in the original utterance exist, they are still kept in one utterance after pre-processing. In each continual learning phase, all training data of one domain are used to train the NLG model, as illustrated in Figure 4.1. Similar pre-processing is done at the granularity of DA intents for experiments in Section 4.4.5. The statistics of the pre-processed MultiWOZ dataset is illustrated in Figure 4.2. The resulting datasets and the pre-processing scripts are open-sourced.

Evaluation metrics. As in Chapter 2, we use the slot error rate (SER) and the BLEU-4 score (Papineni et al., 2002) as evaluation metrics. SER is the ratio of the number of *missing* and *redundant* slots in a generated utterance to the total number of ground truth slots in the DA. To better evaluate the continual learning ability, we use

[3]extracted for NLG at https://github.com/andy194673/nlg-sclstm-multiwoz

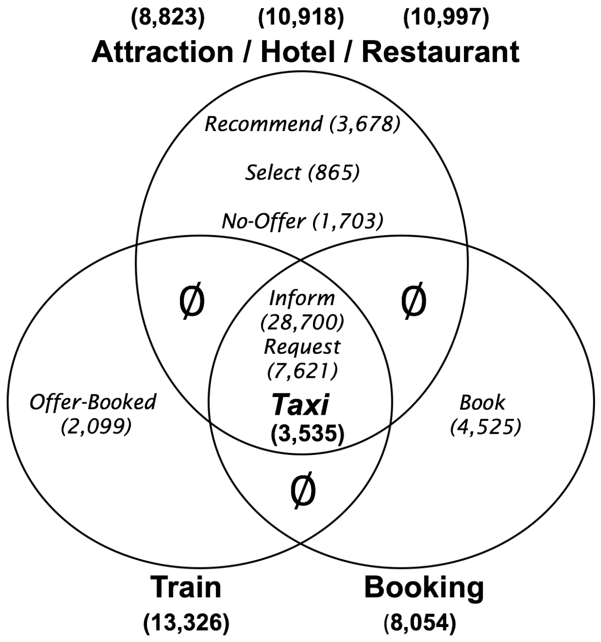


Figure 4.2 – Venn diagram visualizing intents in different domains. The number of utterances of each domain (bold) and intents (italic) is indicated in parentheses.

two additional commonly used metrics (Kemker et al., 2018) for both SER and BLEU-4:

$$\Omega_{all} = \frac{1}{T}\sum_{i=1}^{T}\Omega_{all,i}, \quad \Omega_{first} = \frac{1}{T}\sum_{i=1}^{T}\Omega_{first,i}$$

where T is the total number of continual learning phases; $\Omega_{all,i}$ is the test performance on *all* the phases after the i^{th} phase has been learned; $\Omega_{first,i}$ is that on the *first* phase after the i^{th} phase has been learned. Since Ω can be either SER or BLEU-4, both Ω_{all} and Ω_{first} have two versions. Ω_{all} evaluates the overall performance, while Ω_{first} evaluates the ability to alleviate catastrophic forgetting.

4.4.2 Baseline Methods

Two methods without exemplars are as below:

- ***Finetune***: At each phase, the NLG model is initialized with the model obtained till the last phase, and then fine-tuned with the data from the current phase.
- ***Full***: At each phase, the NLG model is trained with data from the current and *all* historical phases. This is the "upper bound" for continual learning w.r.t. Ω_{all}.

Several exemplar replay (*ER*) methods trained with Eq. (4.4) using different exemplar selection schemes are compared:

- ***ER***$_{herding}$ (Welling, 2009; Rebuffi et al., 2017): This scheme chooses exemplars that best approximate the mean DA vector over all training examples of this phase.
- ***ER***$_{random}$: This scheme selects exemplars at random. Despite its simplicity, the distribution of the selected exemplars is the same as the distribution of the current phase in expectation.
- ***ER***$_{prio}$: The proposed prioritized scheme (c.f. Algorithm 1) to select representative and diverse exemplars.

Based on *ER*$_{prio}$, four regularization methods (including ours) to further alleviate catastrophic forgetting are compared:

- ***L2***: A static L2 regularization by setting $F_i = 1$ in Eq. (4.7). It regularizes all parameters equally.
- ***KD*** (Rebuffi et al., 2017; Wu et al., 2019c; Hou et al., 2019): The widely-used knowledge distillation (KD) loss (Hinton et al., 2015) is adopted by distilling the prediction logit of current model w.r.t. the prediction logit of the model trained till the last phase.
- ***Dropout*** (Mirzadeh et al., 2020): Dropout Hinton et al. (2012) is recently shown by (Mirzadeh et al., 2020) that it effectively alleviates catastrophic forgetting. We tune different dropout rates assigned to the non-recurrent connections.
- ***ARPER*** (c.f. Algorithm 2): The proposed method using adaptive EWC with *ER*$_{prio}$.

We utilize the well-recognized semantically-conditioned LSTM (SCLSTM Wen et al., 2015b) as the base NLG model f_θ [4] with one hidden layer of size 128. Dropout is not used by default, which is evaluated as a particular regularization technique (c.f. *ER*$_{prio}$+*Dropout*). For all the above methods, the learning rate of Adam is set to 5e-3, the batch size is set to 128, and the maximum number of epochs used to train each phase is set to 100. Early stop to avoid over-fitting is adopted when the validation loss does not decrease for 10 consecutive epochs. To fairly compare different methods, they are trained with the identical configuration on the first phase to have a consistent starting point.

[4] Comparisons based on other base NLG models are included in Section 4.4.6.

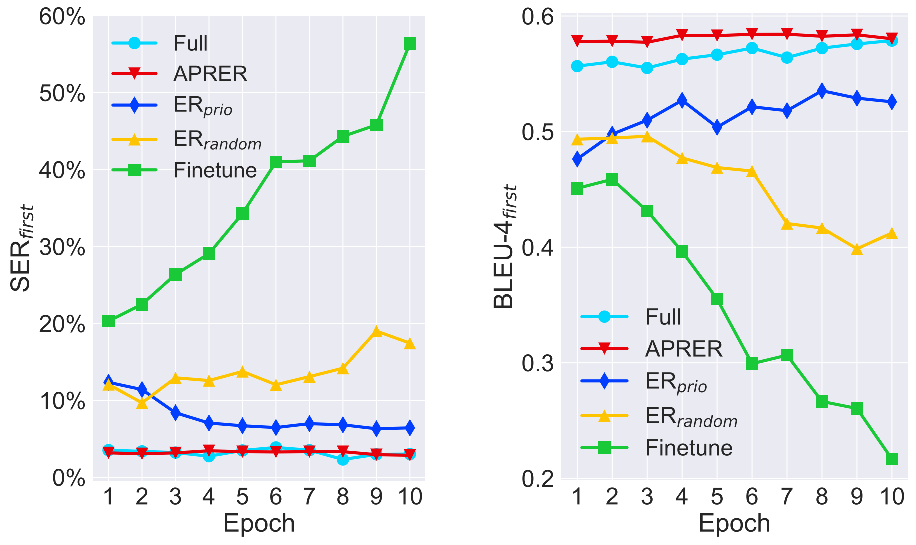


Figure 4.3 – Diagnose the catastrophic forgetting issue in NLG. SER (**Left**) and BLEU-4 (**Right**) on the test data of "*Attraction*" at different epochs when a model pre-trained on the "*Attraction*" domain is continually trained on another "*Train*" domain.

4.4.3 Diagnose Catastrophic Forgetting in NLG

Before proceeding to our main results, we first diagnose whether the catastrophic forgetting issue exists when training an NLG model continually. As an example, a model pre-trained on the "*Attraction*" domain is continually trained on the "*Train*" domain. We present test performance on "*Attraction*" at different epochs in Figure 4.3 with 250 exemplars.

We can observe: (1) catastrophic forgetting indeed exists as indicated by the sharp performance drop of *Finetune*; (2) replaying carefully chosen exemplars helps to alleviate catastrophic forgetting by a large degree, and ER_{prio} does a better job than ER_{random}; and (3) *ARPER* greatly mitigates catastrophic forgetting by achieving similar or even better performance compared to *Full*.

4.4.4 Continual Learning New Domains

In this experiment, the data from six domains are presented sequentially. We test 6 runs with different domain order permutations. Each domain is selected as the first phase for one time, and the remaining five domains are randomly ordered. Results averaged over 6 runs using 250 and 500 total exemplars are presented in Table 4.1. Several interesting observations can be noted:

- All methods except *Finetune* perform worse on all seen phases (Ω_{all}) than on the

	250 exemplars in total				**500 exemplars in total**			
	Ω_{all}		Ω_{first}		Ω_{all}		Ω_{first}	
	SER%	BLEU-4	SER%	BLEU-4	SER%	BLEU-4	SER%	BLEU-4
Finetune	64.46	0.361	107.27	0.253	64.46	0.361	107.27	0.253
$ER_{herding}$	16.89	0.535	9.89	0.532	12.25	0.555	4.53	0.568
ER_{random}	10.93	0.552	6.96	0.553	8.36	0.569	4.41	0.572
ER_{prio}	9.67**	0.578	5.28**	0.578	7.48**	0.597	3.59*	0.620
ER_{prio}+*L2*	14.94	0.579	5.31**	0.587	10.51	0.596	4.28**	0.605
ER_{prio}+*KD*	8.65**	0.586	6.87	0.601	7.37**	0.596	4.89	0.617
ER_{prio}+*Dropout*	7.15**	0.588	5.53**	0.594	6.09*	0.595	4.51**	0.616
ARPER	**5.22**	**0.590**	**2.99**	**0.624**	**5.12**	**0.598**	**2.81**	**0.627**
Full	4.26	0.599	3.60	0.616	4.26	0.599	3.60	0.616

Table 4.1 – Average performance of continually learning 6 domains using 250/500 exemplars. Best Performance excluding "*Full*" are in bold in each column. In each column , * indicates $p < 0.05$ and ** indicates $p < 0.01$ for a one-tailed t-test comparing *ARPER* to the three top-performing competitors except *Full*.

first phase (Ω_{first}). This is due to the diverse knowledge among different phases, which increases the difficulty of handling all the phases. *Finetune* performs poorly in both metrics because of the detrimental catastrophic forgetting issue.

- Replaying exemplars helps to alleviate the catastrophic forgetting issue. Three *ER* methods substantially outperform *Finetune*. Moreover, the proposed prioritized exemplar selection scheme is effective, indicated by the superior performance of ER_{prio} over $ER_{herding}$ and ER_{random}.

- *ARPER* significantly outperforms three *ER* methods and other regularization-based baselines. Compared to the three closest competitors, *ARPER* is significantly better with p-value < 0.05 w.r.t SER.

- The improvement margin of *ARPER* is significant w.r.t SER that is critical for measuring an output's fidelity to a given dialog act. Different methods demonstrate similar performance w.r.t BLEU-4, where several of them approach *Full*, thus are very close to the upper bound performance.

- *ARPER* achieves comparable performance w.r.t to the upper bound (*Full*) on all seen phases (Ω_{all}) even with a very limited number of exemplars. Moreover, it outperforms *Full* on the first phase (Ω_{first}), indicating that *ARPER* better mitigates forgetting the first phase than *Full*, and the latter is still interfered by data in later domains.

	Ω_{all}		Ω_{first}	
	SER%	BLEU-4	SER%	BLEU-4
Finetune	49.94	0.382	44.00	0.375
$ER_{herding}$	13.96	0.542	8.50	0.545
ER_{random}	8.58	0.626	5.53	0.618
ER_{prio}	8.21	0.684	5.20	0.669
ER_{prio}+*L2*	6.87	0.693	4.92	0.661
ER_{prio}+*KD*	10.59	0.664	10.87	0.649
ER_{prio}+*Dropout*	6.32	0.689	5.55	0.658
ARPER	**3.63**	**0.701**	**3.52**	**0.685**
Full	3.08	0.694	2.98	0.672

Table 4.2 – Performance of continually learning 7 DA intents using 250 exemplars. Best Performance excluding "*Full*" are in bold.

	SCVAE		GPT-2	
	Ω_{all}	Ω_{first}	Ω_{all}	Ω_{first}
Finetune	60.83	98.86	28.69	31.76
$ER_{herding}$	17.95	11.48	11.95	10.48
ER_{random}	9.31	7.52	9.87	8.85
ER_{prio}	8.92	6.16	8.72	8.20
ER_{prio}+*L2*	12.47	6.67	10.51	9.20
ER_{prio}+*KD*	6.32	6.09	8.41	8.09
ER_{prio}+*Dropout*	8.01	8.77	7.60	7.72
ARPER	**4.45**	**4.04**	**5.32**	**5.05**
Full	3.99	4.03	4.75	4.53

Table 4.3 – SER in % of using SCVAE and GPT-2 as f_θ. Best Performance excluding "*Full*" are in bold.

4.4.5 Continual Learning New DA Intents

It is also essential for a phase-oriented dialog system to continually learn new functionalities, namely, supporting new DA intents. To test this ability, the data of seven DA intents are presented sequentially in the order of decreasing data size, i.e., *"Inform, Request, Book, Recommend, Offer-Booked, No-Offer, Select"*. Results using 250 exemplars are presented in Table 4.2. We can observe that *ARPER* still largely outperforms other methods, and similar observations for *ARPER* can be made as before. Therefore, we conclude that *ARPER* is able to learn new functionalities continually. Compared to previous experiments, the performance of ER_{prio}+*KD* degrades, while the performance of ER_{prio}+*L2* improves due to the very large data size in the first phase ("*Inform*"), which means that they are sensitive to phase orders.

	Ω_{all}		Ω_{first}	
	SER%	BLEU-4	SER%	BLEU-4
ARPER	4.82	0.592	3.88	0.569
w/o ER	6.41	0.584	5.85	0.559
w/o PE	5.53	0.587	5.85	0.562
w/o AR	5.57	0.587	4.57	0.563

Table 4.4 – Ablation study for *ARPER*. ER / PE / AR stands for the Exemplar Replay loss / Prioritized Exemplars / Adaptive Regularization, respectively.

4.4.6 Results using Different NLG Models

In this experiment, we change the base NLG model From SCLSTM to SCVAE (Tseng et al., 2018) and GPT-2 (Radford et al., 2019). Hyper-parameters of SCVAE are set by default according to https://github.com/andy194673/nlg-scvae, except that the learning rate is set to 2e-3. For GPT-2, we used the pre-trained model with 12 layers and 117M parameters. As in Peng et al. (2020b), exact slot values are not replaced by special placeholders during training as in SCLSTM and SCVAE. We pre-process the dialog act $\mathbf{d}$ into the format of : $\mathbf{d'} = [\ \mathbf{I}\ (\ s_1 = v_1, \ldots, s_p = v_p\ )\ ]$, and the corresponding utterance $\mathbf{Y}$ is appended to be $\mathbf{Y'}$ with a special start token [BOS] and an end token [EOS]. $\mathbf{d'}$ and $\mathbf{Y'}$ are concatenated before feeding into GPT-2. The learning rate of Adam optimizer is set to 5e-5 without weight decay. As GPT-2 converges faster, we train maximum 10 epochs for each phase with early stop applied to 3 consecutive epochs.

Results of using 250 exemplars to continually learn 6 domains starting with "*Attraction*" are presented in Table 4.3. Thanks to the large-scale pre-trained language model, GPT-2 suffers less from the catastrophic forgetting issue because of the better performance of *Finetune*. In general, the relative performance patterns of different methods are similar to those we observed in Section 4.4.4 and 4.4.5. Therefore, we can claim that the superior performance of *ARPER* can generalize to different base NLG models.

4.4.7 In-depth Analysis

Ablation study. In Table 4.4, we compare several simplified versions of *ARPER* to understand the effects of different components. Comparisons are based on continually learning 6 domains staring with "*Attraction*". We can observe that: (1) $\mathcal{L}_{ER}$ is beneficial because dropping it ("w/o ER") degrades the performance of *ARPER*. (2) Using prioritized exemplars is advantageous because using random exemplars ("w/o PE") for *ARPER* impairs its performance. (3) Adaptive regularization is also effective, indicated by the superior performance of *ARPER* compared to using fixed regularization weights ("w/o AR").

Recommend (**Addr=regent street**, **Fee=free**, **Name=Downing College**)	
Reference	*[**Downing College**] is my favorite. It's located on [**regent street**] and it's [**free**] to get in.*
ERER_{prio}*+Dropout*	*[**Downing College**] is located in the city and it's located in the [**regent street**]. it's located at located at! it's located in the [**Slot-Hotel-Area**].* (**missing: Fee=fre**)
ARPER	*I would recommend [**Downing College**]. It is located at [**regent street**] and has a entrance fee of [**free**].* (correct)
Recommend (**Area=centre of town**, **Name=saints church**, **Type=architecture destination**)	
Reference	*There is a [**saints church**] that is an [**architecture destination**] in the [**centre of town**], would you like that?*
ER_{prio}*+Dropout*	*I recommend [**saints church**] in the [**centre of town**]. it is a nice. it is a guest house in a in a [**Slot-Restaurant-Food**].* (**missing: Type=architecture destination**)
ARPER	*[**saints church**] is a [**architecture destination**] in the [**centre of town**].* (correct)

Table 4.5 – Sample utterances generated for the first domain ("*Attraction*") after the NLG is continually trained on all 6 domains using 250 exemplars. Redundant and missing slots are colored in orange and blue respectively. Obvious grammar mistakes (redundant repetitions) are colored in purple.

Case study. Table 4.5 shows two examples generated by *ARPER* and the closest competitor (i.e., ER_{prio}*+Dropout*) using SCLSTM on the first domain ("*Attraction*") after the NLG model is continually trained on all 6 domains starting with "*Attraction*". In both examples, ER_{prio}*+Dropout* fails to generate slot "Fee" or "Type", instead, it mistakenly generates slots belonging to later domains ("*Hotel*" or "*Restaurant*") with several obvious redundant repetitions colored in purple. It means that the NLG model is *interfered* by utterance patterns in later domains, and it forgets some old patterns it has learned before. In contrast, *ARPER* succeeds in both cases without forgetting previously learned patterns.

Flow of parameters update. To further understand the superior performance of *ARPER*, we investigate the update of parameters throughout the continual learning process. Specifically, we compare SCLSTM's hidden layer weights obtained from consecutive phases, and the pairwise $\mathcal{L}_1$ difference of two sample transitions is shown in Figure 4.4.

We can observe that ER_{prio}*+Dropout* tends to update almost all parameters, while *ARPER* only updates a small fraction of them. Furthermore, *ARPER* has different sets of important parameters for distinct phases, indicated by different high-temperature areas in distinct weight updating heat maps. In comparison, parameters of ER_{prio}*+Dropout* seem to be updated uniformly in different phase transitions. The above observations

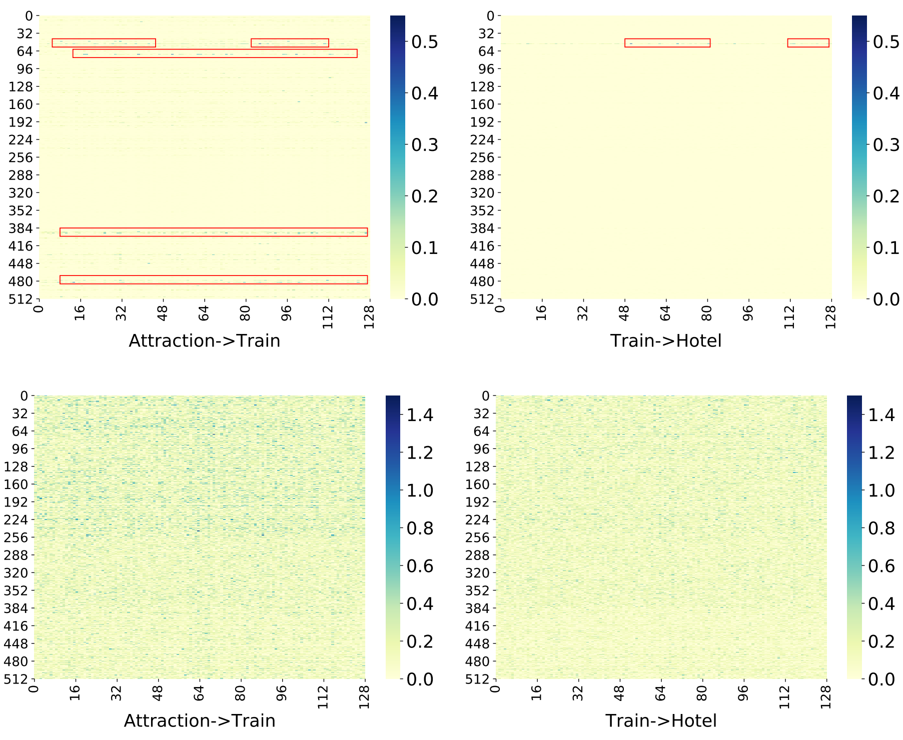


Figure 4.4 – An visualization of the change of SCLSTM's hidden layer weights obtained from two consecutive phases of *ARPER* (**Top**) and *ER*$_{prio}$+*Dropout* (**Bottom**). Two sample phase transitions ("from *Attraction*" to "*Train*", and then from "*Train*" to "*Hotel*") are shown. High temperature areas of *ARPER* is highlighted by red bounding boxes for better visualization.

verify that *ARPER* indeed elastically allocates different network parameters to distinct NLG phases to mitigate catastrophic forgetting.

4.5 Chapter Summary

In this chapter, we study the practical continual learning setting of the natural language generation module in ToD systems. To alleviate catastrophic forgetting, we present *ARPER* which replays representative and diverse exemplars selected in a prioritized manner, and it employs an adaptive regularization term based on EWC (Elastic Weight Consolidation). Extensive experiments on MultiWOZ in different continual learning scenarios reveal the superior performance of *ARPER* . In the next chapter, we formulate and study a similar continual learning setup for recommendation systems. The main setup difference from this chapter is that the evaluation is w.r.t. future observation instead of w.r.t. all historical data.

5 Continual Learning with Knowledge Retention in Recommendation Systems

5.1 Introduction

Due to new privacy regulations that prohibit building user preference models from historical user data, utilizing anonymous short-term interaction data within a browser session becomes popular. Therefore, session-based Recommendation (SR) is increasingly used in real-life online systems, such as E-commerce and social media. The goal of SR is to make recommendations based on user behavior obtained in short web browser sessions. The task is to predict the user's next actions, such as clicks, views, and even purchases, based on previous activities in the same session.

Despite the recent success of various neural approaches (Hidasi et al., 2016; Li et al., 2017; Liu et al., 2018; Kang and McAuley, 2018; Sun et al., 2019), they are developed in an *offline* manner, in which the recommender is trained on a very large static training set and evaluated on a very restrictive testing set in a *one-time* process. However, this setup does not reflect the realistic use cases of online recommendation systems. In reality, a recommendation model needs to be periodically updated with new data steaming in, and the updated model is supposed to provide recommendations for user activities before the next update. In this chapter, we propose a continual learning setup to consider such realistic recommendation scenarios.

In this chapter, we formulate the continual learning setting for the session-based recommendation task to simulate the realistic use cases of training a recommendation model continually. To be specific, at an update cycle [1] t, the recommendation model $f(\theta_{t-1})$ obtained until the last update cycle $t-1$ needs to be updated with new incoming data D_t. After $f(\theta_{t-1})$ is trained on D_t, the updated model $f(\theta_t)$ is evaluated w.r.t. the

This chapter is based on the paper (Mi et al., 2020c) published in the ACM Conference on Recommender Systems (RecSys, 2020). This paper also wins the "**Best Short Paper Award**" at this conference.

[1] An "update cycle" describes a similar notion as the "phase" in Chapter 4, and "update cycle" is a more common and intuitive term in recommendation literature.

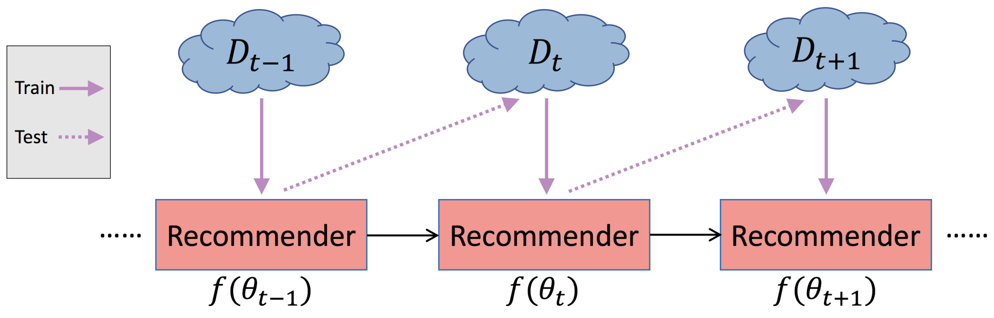


Figure 5.1 – An visualization of the continual learning setup. At each update cycle t, the model is trained with data D_t, and the updated model $f(\theta_t)$ is evaluated w.r.t. to data D_{t+1} before the next model update.

incoming data D_{t+1} before the next update cycle $t+1$. A visualization of the continual learning setup is illustrated in Fig. 5.1, where a recommendation model is continually trained and tested upon receiving data in sequential update cycles.

The major challenge of continual learning is *catastrophic forgetting* (McCloskey and Cohen, 1989; French and Chater, 2002). That is, a neural model updated on new data distributions tends to forget old distributions it has learned before. A naive solution is to retrain the model using all historical data every time. However, it suffers from severe computation and storage overhead in large-scale recommendation applications.

To this end, we propose storing a small set of representative sequences from previous data, namely *exemplars*, and replay them each time the recommendation model needs to be trained on new data. Methods using exemplars have shown great success in different continual learning (Rebuffi et al., 2017; Castro et al., 2018) and reinforcement learning (Schaul et al., 2016; Andrychowicz et al., 2017) tasks. In this chapter, we propose to select representative exemplars of an item using an *herding* technique (Welling, 2009; Rebuffi et al., 2017), and its exemplar size is proportional to the item frequency in the near past. To enforce a stronger constraint on not forgetting previous user preferences, we propose a regularization method based on the well-known knowledge distillation technique (Hinton et al., 2015). We propose to apply a distillation loss on the selected exemplars to preserve the model's knowledge. The distillation loss is further adaptively interpolated with the regular cross-entropy loss on the new data by considering the difference between new data and old ones to deal with different new data distributions flexibly.

Altogether, (1) we are the first to study the practical continual learning setting for the session-based recommendation task; (2) we propose a method called Adaptively Distilled Exemplar Replay (*ADER*) for this task, and benchmark it with state-of-the-art continual learning techniques; (3) experiment results on two widely used datasets empirically demonstrate the superior performance of *ADER* and its ability to mitigate catastrophic forgetting.[2]

5.2 Related Work

Literature review of general continual learning techniques to alleviate the catastrophic forgetting issue is reviewed in Section 4.2. In this section, we mainly review some related studies for the task of session-based recommendation.

Matrix factorization (MF) (Koren et al., 2009) is a general and classical approach to recommendation systems. For session-based recommendation, the interaction matrix to be decomposed is constructed from implicit sequential user feedback. Rendle et al. (2010) proposed a model called Factorizing Personalized Markov Chains (FPMC) to factorize the transitions in Markov chains with low-rank representation for basket recommendations. Factored Item Similarity Models (FISM) (Kabbur et al., 2013) are based on factorizing item-item co-occurrence statistics. Later, He and McAuley (2016) proposed a model called FOSSIL to augment FISM with factorized Markov chains to incorporate sequential information. Recently, Session-based Matrix Factorization (SMF) was proposed by Ludewig and Jannach (2018) for session-based recommendation tasks using session latent vectors. They showed that SMF consistently outperforms previous MF-based methods for SRS. However, MF-based methods are costly to train and update in terms of both computation and storage. For example, Li et al. (2017); Wu et al. (2019b) reported that even 120GB memory is not enough to train FPMC. Therefore, they are, in principle, not suitable for incremental recommendation settings.

Session-based recommendation (SR) can be formulated as a sequence learning problem to be solved by recurrent neural networks (RNNs). The first work (GRU4Rec, Hidasi et al. (2016)) used a gated recurrent unit (GRU) to learn session representations from previous clicks. Based on GRU4Rec, Hidasi and Karatzoglou (2018a) proposed new ranking losses on relevant sessions, and Tan et al. (2016) proposed to augment training data. Attention operation was first used by NARM (Li et al., 2017) to pay attention to specific parts of the sequence. Base on NARM, Liu et al. (2018) proposed STAMP to model users' general and short-term interests using two separate attention operations, and Ren et al. (2019) proposed RepeatNet to predict repetitive actions in a session. Recently, Wu et al. (2019c); Zheng et al. (2019) used graph attention to capture complex transitions of items. Motivated by the recent success of *Tansformer* (Vaswani et al., 2017) and *BERT* (Devlin et al., 2019a) for language model tasks, Kang and McAuley (2018) proposed SASRec using *Transformer*, and Sun et al. (2019) proposed BERT4Rec to model bi-directional information. Despite the broad exploration and success, the above methods were all studied in a static and offline manner as in the classic machine learning paradigm.

[2]Code is available at: https://github.com/DoubleMuL/ADER

5.3 Methodology - ADER

This section introduces some background of neural session-based recommenders in Section 5.3.1. In Section 5.3.2, we propose our method called "Adaptively Distilled Exemplar Replay" (*ADER*) for the continual recommendation task.

5.3.1 Background on Neural Session-based Recommenders

A user *action* in SR is a click or view on an item, and the task is to predict the next user action based on a sequence of user actions in the current web-browser session. Existing neural models $f(\theta)$ typically contain two modules: a **feature extractor** $\phi(\mathbf{x})$ to compute a compact *sequence representation* of the sequence $\mathbf{x}$ of previous user actions, and an **output layer** $\omega(\phi(\mathbf{x}))$ to predict the next user action. Various recurrent neural networks (Hidasi et al., 2016; Hidasi and Karatzoglou, 2018a) and attention mechanisms (Li et al., 2017; Liu et al., 2018; Kang and McAuley, 2018) have been proposed for ϕ, and the common choices for the output layer ω is fully-connect layers(Hidasi et al., 2016) or bi-linear decoders (Li et al., 2017; Kang and McAuley, 2018). In this chapter, we base our comparison on SASRec (Kang and McAuley, 2018), and we refer readers to model details in the original paper to avoid verbosity. Nevertheless, the techniques proposed and compared in this chapter are agnostic to $f(\theta)$. Therefore, a more thorough comparison using different $f(\theta)$ are left for interesting future work.

5.3.2 Adaptively Distilled Exemplar Replay (*ADER*)

In this section, we introduce the proposed method called *ADER*with exemplar replay and adaptive distillation.

Exemplar Replay

To alleviate the widely-recognized catastrophic forgetting issue in continual learning, the model needs to preserve old knowledge it has learned before. To this end, we propose to store past samples, a.k.a *exemplars*, and replay them periodically to preserve previous knowledge. To maintain a manageable memory footprint, we only store a fixed total number of exemplars throughout the entire continual learning process. Two decisions need to be made at each cycle t: (1). how many exemplars should be stored for each item/label? (2). what is the criterion for selecting exemplars of an item/label?

First, we design the number of exemplars of each appeared item in I_t (i.e., the set of appeared items until cycle t) to be proportional to its appearance frequency. In other words, more frequent and popular items contribute a larger portion of selected exemplars to be replayed to the next cycle. Suppose we store N exemplars in total, the number of

Algorithm 6 *ADER*: ExemplarSelection at cycle t

```
Input: $\mathcal{S} = D_t \cup E_{t-1}$; $M_t = [m_1, m_2, ..., m_{|I_t|}]$
1: for $y = 1, ..., |I_t|$ do    // for all seen items
2:     $\mathcal{P}_y \leftarrow \{\mathbf{x} : \forall(\mathbf{x}, y) \in \mathcal{S}\}$
3:     $\mu \leftarrow \frac{1}{|\mathcal{P}_y|} \sum_{\mathbf{x} \in \mathcal{P}_y} \phi(\mathbf{x})$    // compute the mean representation of item $y$
4:     for $k = 1, ..., m_y$ do
5:         $\mathbf{x}^k \leftarrow \arg\min_{\mathbf{x} \in \mathcal{P}_y} \|\mu - \frac{1}{k}[\phi(\mathbf{x}) + \sum_{j=1}^{k-1} \phi(\mathbf{x}^j)]\|$    // select $m_y$ exemplars for $y$ by herding
6:     end for
7:     $E_y \leftarrow \{(\mathbf{x}^1, y), ..., (\mathbf{x}^{m_y}, y)\}$
8: end for
Output: exemplar set $E_t = \cup_{y=1}^{|I_t|} E_y$
```

exemplars $m_{t,i}$ at cycle t for an item $i \in I_t$ is:

$$m_{t,i} = N \cdot \frac{|\{\mathbf{x}, y = i\} \in D_t \cup E_{t-1}|}{|D_t \cup E_{t-1}|}, \tag{5.1}$$

where the second term is the probability that item i appears in the current update cycle, as well as in the exemplars E_{t-1} we kept from the last cycle.

Therefore, the exemplar sizes of different items to be select in the cycle t can be encoded as a vector $M_t = [m_1, m_2, ..., m_{|I_t|}]$.

Second, we need to decide which samples to select as exemplars for each item. We propose to use a herding technique (Welling, 2009; Rebuffi et al., 2017) to select the most representative sequences of an item in an iterative manner based on the distance to the mean feature vector of the item. In each iteration, one sample from $D_t \cup E_{t-1}$ that best approximates the average feature vector (μ) over all training examples of this item (y) is selected to E_t. The details are presented in Algorithm 6.

Adaptive Distillation on Exemplars

The number of exemplars should be reasonably small to reduce memory overhead. As a consequence, the constraint to prevent the recommender from forgetting previous user preference patterns is not strong enough. To enforce a stronger constrain on not forgetting old user preference patterns, we propose to use a knowledge distillation loss (Hinton et al., 2015) on exemplars to consolidate old knowledge better.

At a cycle t, the set of exemplars to be replayed is E_{t-1} and the set of items till the last cycle is I_{t-1}, the proposed knowledge distillation (KD) loss is written as:

$$\mathcal{L}_{KD}(\theta_t) = -\frac{1}{|E_{t-1}|} \sum_{(\mathbf{x},y) \in E_{t-1}} \sum_{i=1}^{|I_{t-1}|} \hat{p}_i \cdot log(p_i), \tag{5.2}$$

Algorithm 7 *ADER*: UpdateModel at cycle t

```
Input: Training data at cycle $t$: $D_t$, Exemplar set until cycle $t-1$: $E_{t-1}$, Set of items
    till cycle $t-1$: $I_{t-1}$, Set of items till cycle $t$: $I_t$
1: Initialize $\theta_t$ with $\theta_{t-1}$
2: while $\theta_t$ not converged do
3:     Train $\theta_t$ with loss in Eq. (5.4)
4: end while
5: Compute $M_t$ using Eq. (5.1)
6: Compute $E_t$ using Algorithm 6 with $\theta_t$ and $M_t$
Output: updated $\theta_t$ and new exemplar set $E_t$
```

where $[\hat{p}_1, \ldots, \hat{p}_{|I_{t-1}|}]$ is predicted distribution (softmax of logits) over I_{t-1} generated by $f(\theta_{t-1})$, and $[p_1, \ldots, p_{|I_{t-1}|}]$ is the prediction of $f(\theta_t)$ over I_{t-1}. $\mathcal{L}_{KD}$ measures the difference between the previous model's outputs and the current model on exemplars, and the idea is to penalize prediction changes on items in previous update cycles.

L_{KD} defined above is interpolated with a regular cross-entropy (CE) loss computed w.r.t. D_t defined below:

$$\mathcal{L}_{CE}(\theta_t) = -\frac{1}{|D_t|} \sum\nolimits_{(\mathbf{x},y)\in D_t} \sum\nolimits_{i=1}^{|I_t|} \delta_{i=y} \cdot log(p_i), \tag{5.3}$$

In practice, the size of incoming data and the number of new items varies in different cycles, therefore, the degree of need to preserve old knowledge varies. To this end, we propose an adaptive weight λ_t to combine $\mathcal{L}_{KD}$ with $\mathcal{L}_{CE}$:

$$\mathcal{L}_{ADER} = \mathcal{L}_{CE} + \lambda_t \cdot \mathcal{L}_{KD}, \quad \lambda_t = \lambda_{base}\sqrt{\frac{|I_{t-1}|}{|I_t|} \cdot \frac{|E_{t-1}|}{|D_t|}} \tag{5.4}$$

In general, λ_t increases when the ratio of the number of old items to that of new items increases, and when the ratio of the exemplar size to the current data size increases. The idea is to rely more on L_{KD} when the new cycle contains fewer new items or fewer data to be learned. The overall training procedure for *ADER* is summarized in Algorithm 7.

5.4 Evaluation

5.4.1 Datasets and Evaluation Metrics

Two widely used datasets are adopted: (1). **DIGINETICA** contains click-streams data on an e-commerce site over five months, and it is used for CIKM Cup 2016 (http://cikm2016.cs.iupui.edu/cikm-cup). (2). **YOOCHOOSE** is another dataset used by RecSys Challenge 2015 (http://2015.recsyschallenge.com/challenge.html) for predicting click-streams on another e-commerce site for over six months.

DIGINETICA	week	0	1	2	3	4	5	6	7	8
	total actions	70,739	37,586	31,089	32,687	30,419	57,913	52,225	57,100	69,042
	new actions	100.00%	18.25%	13.26%	11.29%	10.12%	9.08%	6.64%	6.35%	5.42%
	week	9	10	11	12	13	14	15	16	**Total**
	total actions	82,834	82,935	50,037	63,133	70,050	71,670	56,959	77,065	993,483
	new actions	5.22%	3.02%	3.01%	1.78%	1.83%	0.78%	0.45%	0.27%	/
YOOCHOOSE	day	0	1	2	3	4	5	6	7	8
	total actions	219,389	209,219	218,162	162,637	177,943	307,603	232,887	178,076	199,615
	new actions	100.00%	3.04%	1.74%	1.29%	0.95%	0.57%	0.50%	1.09%	0.74%
	day	9	10	11	12	13	14	15	16	**Total**
	total actions	179,889	123,750	153,565	300,830	259,673	187,348	154,316	105,676	3,370,578
	new actions	0.81%	1.08%	0.56%	0.56%	0.29%	0.41%	0.38%	0.35%	/

Table 5.1 – Statistics of the two datasets; "new actions" indicates the percentage of actions on new items in this update cycle; week/day 0 is only used for training, while week/day 16 is only used for testing.

As in (Hidasi et al., 2016; Li et al., 2017; Liu et al., 2018; Kang and McAuley, 2018), we remove sessions of length 1 and items that appear less than 5 times. To simulate the continual learning scenario, we split the model update cycle of DIGINETICA by *weeks* and YOOCHOOSE by *days* as its volume is much larger. Different time spans also resemble model update cycles at different granulates. In total, 16 update cycles are used to train the recommender on both datasets continually. 10% of the training data of each update cycle is randomly selected as a validation set. Statistics of split datasets are summarized in Table 5.1. We can see that YOOCHOOSE is less dynamic, indicated by the tiny fraction of actions on new items; that is, old items heavily reappear.

Evaluation metrics. Two commonly used evaluation metrics are used: (1). **HR@k**: The hit rate ratio when the desired item is among the top-k recommended items. It can be interpreted as both precision (Liu et al., 2018; Wu et al., 2019b) and recall (Hidasi et al., 2016; Li et al., 2017; Jannach and Ludewig, 2017) because the task is to predict the single immediate next event. (2). **MRR@k**: HR@k does not consider the order of the items recommended, while MRR@k measures the *mean reciprocal ranks* of the desired item in top-k recommended items. The reciprocal rank of the desired item is the inverse of its rank in top-k recommended items. It equals 1 for first place, $\frac{1}{2}$ for second place, and so on. The reciprocal rank is set to zero if the desired item is not in the top-k items. We report the mean value of these two metrics averaged over all 16 update cycles for easier comparison.

5.4.2 Baseline Methods

Several widely adopted baselines in continual learning literature are compared:

- ***Finetune***: At each cycle, the recommender trained till the last task is trained with the data from the current task.

	DIGINETICA				
	Finetune	*Dropout*	*EWC*	*Joint*	*ADER*
HR@20	47.28%	49.07%	47.66%	50.03%	**50.21**%
HR@10	35.00%	36.53%	35.48%	37.27%	**37.52**%
MRR@20	16.01%	16.86%	16.28%	17.31%	**17.32**%
MRR@10	15.16%	16.00%	15.44%	16.43%	**16.45**%

Table 5.2 – Performance averaged over 16 continual update cycles on DIGINETICA.

	YOOCHOOSE				
	Finetune	*Dropout*	*EWC*	*Joint*	*ADER*
HR@20	71.86%	72.20%	71.91%	72.22%	**72.38%**
HR@10	63.82%	64.15%	63.89%	64.16%	**64.41%**
MRR@20	36.49%	36.60%	36.53%	36.65%	**36.71%**
MRR@10	35.92%	36.03%	35.97%	36.08%	**36.14%**

Table 5.3 – Performance averaged over 16 continual update cycles on YOOCHOOSE.

- ***Dropout*** (Mirzadeh et al., 2020): Dropout (Hinton et al., 2012) is recently found by (Mirzadeh et al., 2020) that it effectively alleviates catastrophic forgetting. We applied dropout to every self-attention and feed-forward layer.

- ***EWC*** (Kirkpatrick et al., 2017): It is a well-known method to alleviate forgetting by regularizing parameters important to previous data estimated by the diagonal of a Fisher information matrix computed w.r.t. exemplars.

- ***ADER*** (c.f. Algorithm 7): The proposed method using adaptively distilled exemplars in each cycle with dropout.

- ***Joint***: In each cycle, the recommender is trained (with dropout) using data from the current and *all* historical cycles. This is a common performance "upper bound" for continual learning.

The above methods are applied on top of the state-of-the-art base SR recommender SASRec (Kang and McAuley, 2018) using 150 hidden units and two stacked self-attention blocks. During continual training, we set the batch size to be 256 on DIGINETICA and 512 on YOOCHOOSE. We use Adam optimizer with a learning rate of 5e-4. A total of 100 epochs are trained, and "early stop" is applied if validation performance (HR@20) does not improve for five consecutive epochs. Other hyper-parameters are tuned to maximize HR@20. The dropout rate of *Dropout*, *ADER*, and *Joint*is set to 0.3; 30,000 exemplars are used by default for *EWC* and *ADER*; λ_{base} of *ADER* is set to 0.8 on DIGINETICA and 1.0 on YOOCHOOSE.

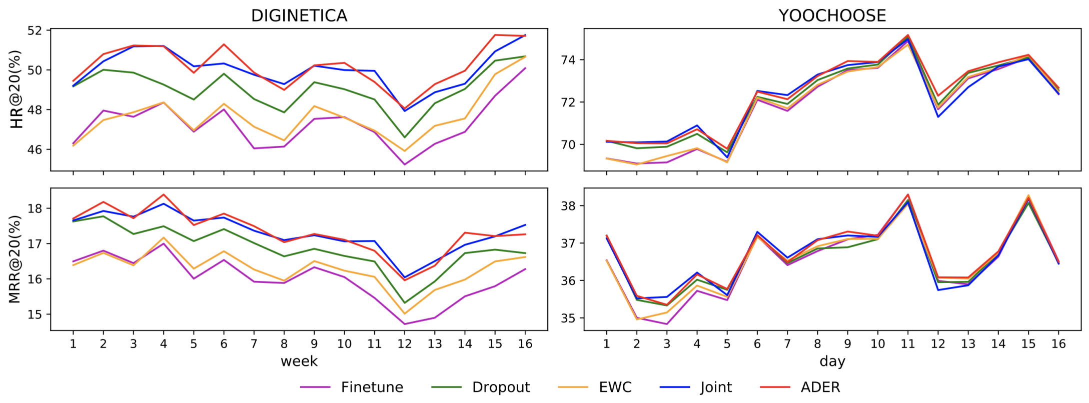


Figure 5.2 – Disentangled HR@20 (Top) and MRR@20 (Bottom) at each continual learning update cycle on two datasets.

5.4.3 Overall Results on Two Datasets

Results averaged over 16 update cycles are presented in Table 5.2 and 5.3, and several interesting observations can be noted:

- *Finetune* already works reasonably well, especially on the less dynamic YOOCHOOSE dataset. The performance gap between *Finetune* and *Joint* is less significant than typical continual learning setups (Rebuffi et al., 2017; Li and Hoiem, 2018; Wu et al., 2019c; Hou et al., 2019). The reason is that catastrophic forgetting is not severe because old items can frequently *reappear* in recommendation tasks and the evaluation is only w.r.t. the next cycle.
- *EWC* only outperforms *Finetune* marginally, and it performs worse than *Dropout*.
- *Dropout* is effective, and it notably outperforms *Finetune*, especially on the more dynamic DIGINETICA dataset.
- *ADER* significantly outperforms other methods, and the improvement margin over other methods is larger on the more dynamic DIGINETICA dataset. Furthermore, it even outperforms *Joint*. This result empirically reveals that *ADER* is a promising solution for the continual recommendation setting by effectively preserving user preference patterns learned before.

Detailed performance at each update cycle is plotted in Figure 5.2. We can see that the advantage of *ADER* is significant on the more dynamic DIGINETICA dataset. On the less dynamic YOOCHOOSE dataset, the gain of *ADER* mainly comes from the more dynamic starting cycles with relatively more actions on new items. Different methods show comparable performance at later stable cycles with few new items, including the vanilla *Finetune*.

	10k	20k	30k
HR@20	49.59%	50.05%	50.21%
HR@10	36.92%	37.40%	37.52%
MRR@20	17.04%	17.29%	17.32%
MRR@10	16.17%	16.42%	16.45%

Table 5.4 – Different exemplar sizes for *ADER*.

	ER_{random}	ER_{loss}	$ER_{herding}$	$ADER_{equal}$	$ADER_{fix}$	$ADER$
HR@20	49.14%	49.31%	49.44%	49.92%	50.09%	50.21%
HR@10	36.61%	36.65%	36.88%	37.21%	37.41%	37.52%
MRR@20	16.79%	16.90%	16.95%	17.23%	17.29%	17.32%
MRR@10	15.92%	16.02%	16.08%	16.35%	16.41%	16.45%

Table 5.5 – Ablation study for *ADER*.

5.4.4 In-depth Analysis

In the following experiments, we conduct an in-depth analysis of the results on the more dynamic DIGINETICA dataset.

Different number of Exemplars

We study the effect of a varying number of exemplars for *ADER*. Besides using 30k exemplars, we test using only 10k/20k exemplars, and results are shown in Table 5.4. We can see that the performance of *ADER* only drops marginally as exemplar size decreases from 30k to 10k. This result reveals that *ADER* is insensitive to the number of exemplars, and it works reasonably well with a smaller number of exemplars.

Ablation Study

In this experiment, we compare *ADER* to several simplified versions to justify our design choices. (1) $ER_{herding}$: A vanilla exemplar replay different from *ADER* by using a regular L_{CE}, rather than L_{KD}, on exemplars. (2) ER_{random}: It differs from $ER_{herding}$ by selecting exemplars of an item at random. (3) ER_{loss}: It differs from $ER_{herding}$ by selecting exemplars of an item with smallest $\mathcal{L}_{CE}$. (4) $ADER_{equal}$: This version differs from *ADER* by selecting an equal number of exemplars for each item, that is, the assumption that more frequent items should be stored more is removed. (5) $ADER_{fix}$: This version differs from *ADER* by not using the adaptive λ_t in Eq. (5.4), but a fixed λ.

Comparison results are presented in Table 5.5, and several observations can be noted: (1) Herding is effective to selected exemplars, indicated by the better performance of $ER_{herding}$ over ER_{random} and ER_{loss}. (2) The distillation loss in Eq. (5.2) is helpful, indicated by the better performance of three versions of *ADER* over three vanilla *ER* methods. (3) Selecting exemplars proportional to item frequency is helpful, indicated by the better performance of *ADER* over $ADER_{equal}$. (4) The adaptive λ_t in Eq. (5.2) is helpful, indicated by the better performance of *ADER* over $ADER_{fix}$.

5.5 Chapter Summary

In this chapter, we study the practical and realistic continual learning setting for session-based recommendation tasks. To prevent the recommender from forgetting user preferences learned before, we propose *ADER* by replaying carefully chosen exemplars from previous cycles and an adaptive distillation loss. Experiment results on two widely used datasets empirically demonstrate the promising performance of *ADER*. For chapters in the next part, we study several initial attempts to deal with both data efficiency and knowledge retention in a unified framework.

Part III

Solving Knowledge Retention and Data Efficiency Together

6 Non-parametric Memorization and Prediction in Recommendation Systems

6.1 Introduction

In many realistic learning scenarios, the demand for data efficiency and knowledge retention is equally important for a machine learning system. For example, in recommendation applications, such as news, forums, e-commerce, and other social media, new items are frequently added with limited observations on new user interests and preferences drift (Koren, 2009). Therefore, recommendations need to be adapted to such changes in a data-efficient manner. At the same time, static preferences on old items still need to be preserved by the model. In this chapter, we study data efficiency and knowledge retention together in the recommendation scenario and propose to use non-parametric methods for both knowledge retention and data-efficient prediction.

In Chapter 5, we study a continual learning setup for the session-based recommendation task to learn from multiple update cycles with proper knowledge retention. However, new items and new user preferences between each model update cycle cannot be properly captured. That is, events of the current cycle are predicted by the model updated until the last cycle. As illustrated in Fig. 6.1, new items and preference distribution shifts in the evaluation phase can not be properly captured.

This chapter studies how to quickly capture the new items and preferences during the model inference phase. To better address the *dynamic* nature of the session-based (*SR*) tasks, we study it from an incremental adaptation [1] perspective, referred to as *incremental session-based recommendation.* In this setting, new items and preferences appear incrementally during the evaluation phase, and models need to incorporate the

This chapter is based on two papers: Mi and Faltings (2017) is published in the International Conference on Educational Data Mining (EDM, 2017), and Mi and Faltings (2020) is published in the International Joint Conference on Artificial Intelligence (IJCAI, 2020).

[1]Some literature use the term "incremental" to denote our continual learning setup in Part II. In this thesis, we clarify that our "incremental adaptation" refers to incrementally capturing new knowledge during model inference/evaluation.

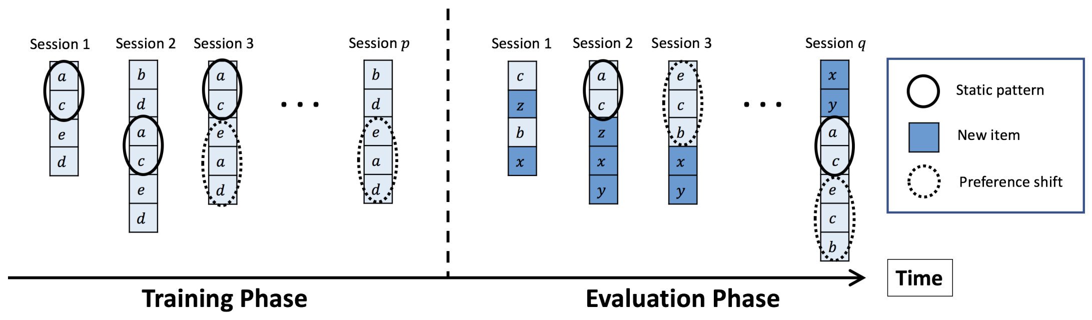


Figure 6.1 – A typical session-based recommendation setting focus on predicting static patterns and ignores new items and preference distribution shifts in the evaluation phase. In realistic scenarios, new items (x, y, z) that are not part of the training phase and preference distribution shifts on old items should be incrementally captured. As a preference shift example, e is often followed by a and d in the training phase but it shifts to $\langle e \rightarrow c \rightarrow b\rangle$ in the evaluation phase.

new preferences incrementally while preserving old ones that are still useful. The setup requires a recommender to be incrementally updated with data observed during the evaluation phase. We summarize two main challenges of using traditional neural networks for incremental *SR* scenarios:

1. *knowledge retention* (McCloskey and Cohen, 1989): incorporating new patterns requires additional training and often reduces performance on old patterns.
2. *data efficiency*: the number of observations on new items and patterns is often small such that models need to capture them quickly with limited observations.

In this chapter, we propose two approaches for the incremental *SR* scenario. In Section 6.3, we propose a non-parametric method, called **NE**ighbor-guided **C**ontext **T**ree (NECT). Compared to parametric models, such as neural networks, that specify a fixed and finite number of model parameters independent of dataset size, non-parametric methods (Härdle and Linton, 1994) assume no distribution of the underlying data as they use an unbounded number of parameters to specify the data. A nice feature is that the amount of information they capture evolves as the amount of data grows. This makes them particularly suitable for incremental scenarios requiring rapid adaptation to new knowledge. NECT uses a context-dependent tree structure (Willems et al., 1995) as a non-parametric memory such that knowledge is maintained and retrieved elegantly by context. A prediction model based on the mixture of experts and a recursive Bayesian weight update procedure is applied within the tree structure. NECT also construct dynamic neighbor-guided candidate sets, improving recommendation accuracy and computation efficiency. NECT can address the two challenges above mentioned before. The hierarchical context tree structure used in NECT helps address both knowledge retention and data efficiency challenges. First, the accommodation of new knowledge is modeled by newly created contexts. Old contexts and their corresponding estimations remain intact such that old knowledge is retained. Second, new items and preference patterns with very limited observations can be immediately captured through newly created contexts to address the data efficiency challenge.

In Section 6.4, we propose a method called **M**emory **A**ugmented **N**eural model (MAN) that augments a neural recommender with a *non-parametric* memory component with key-value pairs to capture new items and preferences incrementally. Similar frameworks are proposed for language model (Merity et al., 2017; Grave et al., 2017a), neural machine translation (Tu et al., 2018), and image recognition (Orhan, 2018) tasks to better capture minority class. A memory prediction is computed from the non-parametric memory component using the k-nearest neighbor (KNN) principle. With a frequently updated non-parametric memory component, the good *memorization* ability (Cohen et al., 2018; Khandelwal et al., 2020) of KNN helps to quickly capture new knowledge with limited observations to improve data efficiency. The predictions of neural and memory components are combined by another lightweight gating network. MAN is *agnostic* to the neural model as long as it learns meaningful sequence representations. Therefore, it can be easily and broadly applied to various neural recommenders. MAN is able to deal with the two challenges of incremental *SR* mentioned above. First, the non-parametric memory component of MAN achieves a long-term memory to remember long histories of observations to mitigate catastrophic forgetting. Second, the memory prediction of MAN helps it to better capture new patterns with a small number of observations to achieve data efficiency.

The contribution of this chapter is summarized as follows:

- We propose to improve both data efficiency and knowledge retention using non-parametric methods. As a case study, we utilize a recommendation scenario that requires both capabilities in practice.

- We propose two usages of non-parametric methods for both memorization and prediction. NECT in Section 6.3 utilizes a hierarchical context tree structure for memorization, and a prediction model is designed on top of it. MAN in Section 6.4 utilizes a memory component with a key-value structure, and a non-parametric prediction based on nearest neighbors is combined with the regular neural prediction.

- We benchmark NECT and MAN with various baselines in the incremental SR scenario, and we demonstrate that they both outperforms state-of-the-art methods by notable margins. We analyze the results and show that it is mainly due to the improved precision on infrequent items without losing precision on frequent ones.

6.2 Related Work

6.2.1 Non-parametric Session-based Recommendation

Methods based on matrix factorization and neural networks for the task of session-based recommendation are included in Section 5.2. Therefore, we mainly review some non-parametric methods for this task in this chapter.

Collaborative filtering (CF) methods based on K-nearest neighbors have been proven to be effective and are widely employed in industry, such as the simplest item-to-item CF approach (Sarwar et al., 2001). However, they are unable to represent sequences. Recently, Jannach and Ludewig (2017) proposed a session-based KNN (SKNN) based on item co-occurrence statistics from different sessions. They show that the model is efficient thanks to pre-computed in-memory statistics, and it achieves state-of-the-art performance. Lately, various scoring functions (Ludewig and Jannach, 2018) were proposed based on SKNN by putting more weight on items that appear later in a session. Recently, Garg et al. (2019) proposed STAN to incorporate time information using decay factors at three different places for determining session-level similarity and item relevance. Nevertheless, these methods do not explicitly consider the sequential item transitions within a session.

Another group of non-parametric methods to predict user's sequential actions is based on Markov chains. Shani et al. (2005) shows that the fixed-order Markov model is not enough for this task as the fixed-order assumption is too strong and limits the recommendation accuracy. Therefore, they consider a finite mixture of Markov models with fixed weights. However, it is still challenging to set the maximum order of the Markov chain and the mixing coefficients. Variable-order Markov model (VMM) (Begleiter et al., 2004) is a class of models that extend Markov chain models such that the order of the Markov chain can be dynamically determined depending on different contexts. The flexibility of VMM turns out to be of advantage for many applications (Begleiter et al., 2004). For the method proposed in Section 6.3, we use a form of VMM using a non-parametric structure called context tree (Willems et al., 1995), which was originally used for lossless data compression. Kozat et al. (2007); Dimitrakakis (2010) applied it to various discrete sequence prediction tasks. Recently, it was applied to news recommendation by Garcin et al. (2013, 2014). However, these non-parametric methods require tuning hyper-parameters, and the incremental adaptation issue remains to be addressed.

6.2.2 Memory Augmented Neural Model

Recently, various memory modules have been proposed to augment neural networks for remembering long-term information (Graves et al., 2014; Sukhbaatar et al., 2015; Grave et al., 2017b,a; Merity et al., 2017) or learning infrequent or rare events (Santoro et al., 2016b; Kaiser et al., 2017; Sprechmann et al., 2018). Typically, a memory component $\mathbf{M}$

is maintained to remember data seen before.

There are many variants of how to read from **M** and mix the entries retrieved from **M** with the network computation. One approach is through some differentiable context-based lookup mechanisms (Graves et al., 2014; Sukhbaatar et al., 2015; Santoro et al., 2016b) for learning to match the current activation to past activations stored in **M**. However, these mechanisms often require strong supervision, and the size of the **M** has to be limited and fixed. Another approach is using a simple mixture model. In this case, a non-parametric prediction is computed based on the similarity between the entries in **M** and the current input. The neural network's prediction is directly interpolated with the non-parametric prediction from **M**. This approach has be shown simple but effective for language modeling (Grave et al., 2017b,a), neural machine translation (Tu et al., 2018), and image classification (Orhan, 2018).

To scale up to the large number of entries that get stored in **M**, efficiently retrieving nearest neighbors from **M** is necessary. Several approximate search methods have been applied, such as locality sensitive hashing by Kaiser et al. (2017) and product quantization by Grave et al. (2017a). Recently, Sprechmann et al. (2018) introduces a framework to use nearest neighbors retrieved from **M** for parameter adaptation during model inference for the fast acquisition of new knowledge.

Our proposed methods in Section 6.4 is inspired by a recently proposed *non-parametric* memory module that is not trained jointly with neural models. Grave et al. (2017b); Merity et al. (2017) introduce a cache to augment RNNs for language modeling task. They later improve this cache to unbounded size (Grave et al., 2017a) and achieve significant performance improvement. A similar memory module is also proposed for neural machine translation tasks (Tu et al., 2018) and image recognition tasks (Orhan, 2018).

6.3 Methodology - NECT

In this section, we first introduce the context tree (CT) data structure. Then, we present the NECT algorithm that uses the CT structure, a recursive Bayesian update procedure, and dynamical neighbor-guided candidate item sets. Afterward, we discuss some properties of NECT.

Before we proceed, we first recap the task of session-based recommendation described in Chapter 5. The task is predicting user's next event y_t from a session $\mathbf{x}_t = \langle x_1, \dots, x_t \rangle$ of events until time t observed in a web browser session. Each event x_k is a click on an item at time k. We use "event" and "item" interchangeably in this chapter.

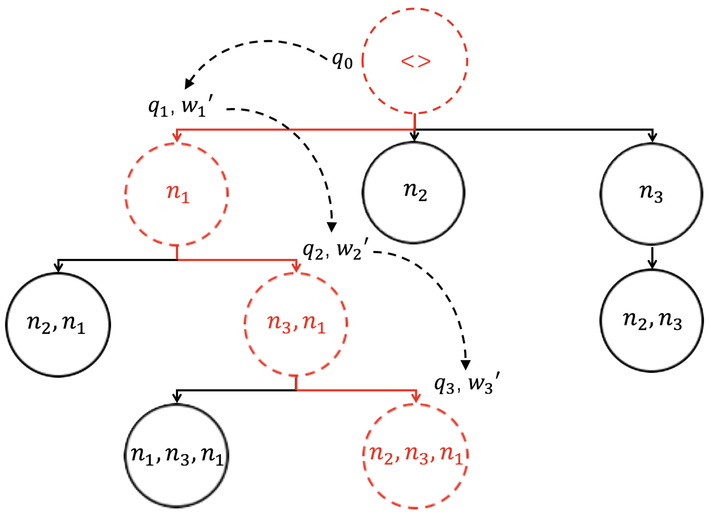


Figure 6.2 – An example context tree with sequences over with candidate item set $\mathbf{N} = \{n_1, n_2, n_3\}$. For the sequence $\mathbf{x}_t = \langle n_2, n_3, n_1 \rangle$, nodes in red-dashed are *active* nodes. The recursive computations in Eq.(6.4) and Eq.(6.5) are indicated by black-dashed arrows.

6.3.1 Context Tree Structure

Before presenting our NECT algorithm, we first define the underlying context tree structure.

Definition 1. *(Suffix) Suppose $\mathbf{x}_t = \langle x_1, \dots, x_t \rangle$ represents a sequence of length t. $\mathbf{x}_k$ is a suffix of $\mathbf{x}_t$ if $k \le t$, and the last k elements of $\mathbf{x}_t$ are the same as $\mathbf{x}_k$. For example, $\langle n_3, n_1 \rangle$ is a suffix of sequence $\langle n_2, n_3, n_1 \rangle$. We write $\mathbf{x}_k \prec \mathbf{x}_t$ when $\mathbf{x}_k$ is a suffix of $\mathbf{x}_t$.*

Definition 2. *(Context Tree) A context tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ is a suffix tree, composed of nodes $\mathcal{V}$ and edges $\mathcal{E}$. A node v_i corresponds to a suffix represented by a sequence $\mathbf{x}_i$. There is an edge e_{ij} from node v_i to v_j if $\mathbf{x}_i \prec \mathbf{x}_j$, and $j = i + 1$.*

Definition 3. *(Suffix Set) Suppose $\mathcal{S}$ denotes the set of all sequences in the training data. For a node v_i with the suffix sequence $\mathbf{x}_i$, we define the suffix set S_i associated with this node to be the set of all possible sequences in $\mathcal{S}$ ending with the suffix $\mathbf{x}_i$. Mathematically, we could write is as $S_i = \{\mathbf{x} \in \mathcal{S} : \mathbf{x}_i \prec \mathbf{x}\}$.*

The session-based recommendation task is predicting the next event y_t by $\mathbb{P}(y_t|\mathbf{x}_t)$ from a session $\mathbf{x}_t = \langle x_1, \dots, x_t \rangle$ of events until time t. First, we identify the context relevant to the input sequence $\mathbf{x}_t$ by *matching* it to the context tree structure defined before.

Definition 4. *(Context and Active Nodes of a Sequence) We define the context $\mathcal{C}(\mathbf{x}_t)$ of $\mathbf{x}_t$ to be all suffixes of $\mathbf{x}_t$: $\mathcal{C}(\mathbf{x}_t) = \{\mathbf{x}_i : \mathbf{x}_i \prec \mathbf{x}_t\}$. The set of nodes corresponds to these* *suffixes are called* active nodes $\mathcal{A}(\mathbf{x}_t) = \{v_i \in \mathcal{V} : \mathbf{x}_i \prec \mathbf{x}_t\}$.

Fig. 6.2 illustrates a simple context tree with sequences over with candidate item set $\mathbf{N} = \{n_1, n_2, n_3\}$. Each node in the tree corresponds to a suffix characterized by a sequence, and it is associated with a suffix set. For instance, the node $\langle n_3, n_1 \rangle$ represents a suffix represented by sequence $\langle n_3, n_1 \rangle$, and the suffix set at this node is the set of sequences that end with suffix $\langle n_3, n_1 \rangle$. As for context matching, the context and the corresponding active nodes of the sequence $\mathbf{x}_t = \langle n_2, n_3, n_1 \rangle$ in Fig. 6.2 are the red-dashed nodes because they are all suffixes of $\mathbf{x}_t$. The context defined in this way covers the complete set of relevant suffixes of $\mathbf{x}_t$ at different **granularity**. For example, the general suffix at depth one captures the last (most recent) event in $\mathbf{x}_t$. As we go deeper along the path of $\mathcal{A}(\mathbf{x}_t)$, we obtain more specific suffixes that match longer suffixes of $\mathbf{x}_t$.

Such hierarchical context tree structure has two properties:

- *The suffix sets at a given depth do not intersect with each other*; for example, the suffix sets at depth one in Fig. 6.2 separate sequences that end with $\langle n_1 \rangle$, $\langle n_2 \rangle$, and $\langle n_3 \rangle$. This property helps to separate sequences ending with different suffixes to different paths.

- *If node v_i is the ancestor of node v_j, then* $\mathbf{x}_i \prec \mathbf{x}_j$; for example, suppose $v_j = \langle n_3, n_1 \rangle$, the sequence at its ancestor $v_i = \langle x_1 \rangle$ is a suffix of $\mathbf{x}_j$. This property allows to model suffix in a hierarchical manner with more specific and longer suffixes obtained at deeper nodes.

In the next section, we introduce how to build a recommendation model using the complete set of *active nodes* $\mathcal{A}(\mathbf{x}_t)$ of $\mathbf{x}_t$.

6.3.2 Local Expert at Each Node

For each node v_i, a local prediction model (expert) is maintained to compute a probability distribution $\mathbb{P}_i$ of the next item to click at this particular local context. The probability $\mathbb{P}_i(y_t = \hat{x})$ that an item $\hat{x}$ is the clicked next depends on the number of times $\alpha_{i\hat{x}}$ that this item $\hat{x}$ has been clicked next when this node is active. $\alpha_{i\hat{x}}$ can be computed from the suffix set S_i of this node that includes all sequences ending with the suffix. As in Eq.(6.1), we use a Dirichlet-multinomial prior for each expert initialized with the initial count α_0, and we set it to one in experiments. The probability is marginalized by the total number of times that different items in the candidate set $\mathbf{N}$ are clicked at this local context. Conceptually, the local expert corresponds to a standard Markov model where the Markov order equals to the depth i of the node.

$$\mathbb{P}_i(y_t = \hat{x}) = \frac{\alpha_{i\hat{x}} + \alpha_0}{\sum_{n \in \mathbf{N}} \alpha_{in} + \alpha_0} \tag{6.1}$$

6.3.3 Combining Predictions from Local Experts

As we mentioned before that $\mathcal{A}(\mathbf{x}_t)$ covers the complete set of relevant suffixes for $\mathbf{x}_t$ at different granularity. To combine all local experts $\mathbb{P}_i$ associated with the *active nodes* in $\mathcal{A}(\mathbf{x}_t)$ to generate recommendations for $\mathbf{x}_t$, a mixture prediction of local experts can be computed as in Eq.(6.2) below:

$$\mathbb{P}(y_t = \hat{x}|\mathbf{x}_t) = \sum_{v_i \in \mathcal{A}(\mathbf{x}_t)} u_i \mathbb{P}_i(y_t = \hat{x}), \tag{6.2}$$

where u_i is the weight assigned to the expert associated with node v_i. Instead of using fixed weights to combine the local predictions (Willems et al., 1995), we adopt a Bayesian approach proposed by (Dimitrakakis, 2010) to learn the weights from data.

A *helper weight* $w_i \in [0, 1]$ is defined for each node v_i as the probability that the prediction is generated by expert of node v_i if it can be generated by the first i active nodes from the root. We can derive the weight $u_i(\mathbf{x}_t)$ as below:

$$u_i = w_i \prod_{j:S_j \subset S_i} (1 - w_j), \tag{6.3}$$

where $\prod_{j:S_j \subset S_i}(1 - w_j)$ computes the probability that the prediction is not generated by nodes deeper than v_i. As we look along the most specific node to the root in $\mathcal{A}(\mathbf{x}_t)$, with probability equal to w_i we use the local expert prediction at node v_i without considering experts at more general suffixes. Therefore, the *combined prediction* q_i from the first i experts from the root can be computed recursively as:

$$q_i(y_t = \hat{x}) = w_i \mathbb{P}_i(y_t = \hat{x}|\mathbf{x}_t) + (1 - w_i) q_{i-1}, (\hat{x}) \tag{6.4}$$

where the local prediction $\mathbb{P}_i$ at the current depth is given the weight w_i, while the former combined prediction is weighted with the remaining $1 - w_i$. For the expert node at depth i, the recursive computation estimates whether it makes a better prediction than the combined prediction $q_{i-1}(\hat{x})$ from shallower active experts. After the recursion from the root to the most specific node in $\mathcal{A}(\mathbf{x}_t)$, q_t is equivalent to $\mathbb{P}(y_t = \hat{x}|\mathbf{x}_t)$ in Eq.(6.2) and we use q_t to generate recommendations for a session $\mathbf{x}_t$.

With the helper weight and the combined prediction presented above, we update the helper weight in closed form by taking into account the success of the prediction of the next item. As item x_{t+1} is clicked next, the helper weight w_i at an active node v_i is updated via Bayes' theorem as:

$$w_i' = \frac{w_i \mathbb{P}_i(y_t = x_{t+1} | \mathbf{x}_t)}{q_i(y_t = x_{t+1})} \tag{6.5}$$

where the likelihood is the local prediction $\mathbb{P}_i$ at this node and it is marginalized by the combined prediction q_i on the desired next item x_{t+1}. More derivations and proofs of this Bayesian update can be found in (Dimitrakakis, 2010). In addition, as the combined prediction q_i is computed recursively, the helper weight is also updated in parallel to q_i on $\mathcal{A}(\mathbf{x}_t)$ from the root to the leave as illustrated by the dashed arrows in Figure 6.2.

6.3.4 Neighbor-guided Candidate Sets

The local prediction $\mathbb{P}_i$ in Eq.(6.1) and the combined prediction q_i in Eq.(6.4) need to be computed over a set of candidate items. Therefore, we to define a dynamic candidate item set $\mathbf{N}_{\mathbf{x}_t}$ for each $\mathbf{x}_t$. We can simply use a fixed $\mathbf{N}_{\mathbf{x}_t}$ by including all possible items. However, it is expensive and redundant to compute scores for a large number of items every time. Moreover, it is not helpful to use a fixed set of candidate items for different sessions when user interests drift in dynamic environments.

We propose to compute candidate sets conditioned on contexts based on item co-occurrence statistics in historical sessions inspired by (Jannach and Ludewig, 2017). We maintain an item co-occurrence matrix $\mathbf{M}$ that keeps track of whether two items appeared together in any sessions in the training data. If so, these two items are *neighbors* of each other. For a particular item x_k, the set $\mathbf{N}_{x_k}$ of its neighbors can be easily computed from the corresponding row vector in $\mathbf{M}$. When we predict the next event y_t given a session $\mathbf{x}_t = \langle x_1, \ldots, x_t \rangle$, the *neighbor-guided candidate set* $\mathbf{N}_{\mathbf{x}_t}$ for $\mathbf{x}_t$ is computed by the *set union* of the neighbors of all items in $\mathbf{x}_t$ as:

$$\mathbf{N}_{\mathbf{x}_t} = \bigcup_{k=1}^{t} \mathbf{N}_{x_k} \tag{6.6}$$

In this way, candidate sets are dynamically formed to handle different types of items/patterns without tuning hyper-parameters. The size of $\mathbf{N}_{\mathbf{x}_t}$ could be large when items in session $\mathbf{x}_t$ are frequently seen in other sessions. If a session mainly contains new or infrequent items, the corresponding $\mathbf{N}_{\mathbf{x}_t}$ will be small. Furthermore, the item co-occurrence matrix $\mathbf{M}$ can be efficiently and continuously updated to include new item co-occurrence relations.

6.3.5 Pipeline of NECT Algorithm

For a typical training data $\{\mathbf{x}_t = \langle x_1, \dots, x_t \rangle, y_t = x_{t+1}\}$ in session-based recommendation tasks. $\mathbf{x}_t$ is the sequence of events till time t, and y_t is the next event. The NECT algorithm presented in Algorithm 8 contains both training and testing steps.

Training procedure. The *training step* is formulated as updating NECT with a single pair of observations $(\mathbf{x}_t, y_t = x_{t+1})$. The set of active nodes $\mathcal{A}(\mathbf{x}_t)$ is computed by matching $\mathbf{x}_t$ to the current context tree $\mathcal{T}$ as described in Definition 4. First, the item co-occurrence matrix $\mathbf{M}$ is update with new co-occurrence relationship between x_{t+1} and all items in $\mathbf{x}_t$. Second, local experts of all active nodes in $\mathcal{A}(\mathbf{x}_t)$ increment the count on x_{t+1} by 1 and helper weights of all active nodes in $\mathcal{A}(\mathbf{x}_t)$ from the root are updated according to Eq.(6.5) recursively based on the next event $y_t = x_{t+1}$. Third, the current tree $\mathcal{T}$ is expanded with new nodes corresponds to all suffixes of the new sequence $\mathbf{x}_{t+1} = \langle \mathbf{x}_t, x_{t+1} \rangle$ that are not in $\mathcal{T}$. As we defined the set of all suffixes of a sequence to be the context of this sequence in Definition 4, we denote the tree expansion as $\mathcal{T}' = \mathcal{T} \bigcup \mathcal{C}(\mathbf{x}_{t+1})$. The helper weight of the newly added node is initialized to one divided by the length of the suffix because we do not trust the initial estimate at a very specific suffix node more than its ancestors.

Testing procedure. During the *recommendation step* for a session $\mathbf{x}_t$, NECT first computes a neighbor-guided candidate set $\mathbf{N}_{\mathbf{x}_t}$ for $\mathbf{x}_t$ as in Eq. (6.6). Then, it matches the current session $\mathbf{x}_t$ to the current context tree $\mathcal{T}$ to identify a set of active nodes $\mathcal{A}(\mathbf{x}_t)$. Lastly, it generates the Top_k recommendation list using the recursive computation though $\mathcal{A}(\mathbf{x}_t)$ from the root in Eq. (6.4) regarding items in $\mathbf{N}_{\mathbf{x}_t}$.

NECT has the following three nice properties

- **NECT is computationally efficient**. both training and recommendation procedures can be computed recursively by going through a single pass of activated nodes (experts) from the root. For a session of events $\mathbf{x}_t$, it takes $\mathcal{O}(|\mathbf{x}_t|)$ operations to update the model parameters and $\mathcal{O}(|\mathbf{x}_t| \times |\mathbf{N}_{\mathbf{x}_t}|)$ operations to recommend for $\mathbf{x}_t$, where $|\mathbf{x}_t|$ is the length of the session and $|\mathbf{N}_{\mathbf{x}_t}|$ is the size of the neighbor-guided candidate set for this session. The space complexity of NECT is bounded by the number of nodes in the context tree, and it depends on the size of the dataset as we do not generate extra nodes unless the suffix occurs in the dataset.
- **NECT is flexible.** To avoid the problem of selecting a fixed order of a Markov chain, NECT uses a flexible structure for estimating a variable-order Markov model such that a higher order is used where necessary. Each local expert corresponds to a Markov model with the Markov order equal to the depth of the node, and NECT computes a dynamic mixture of these local estimations with the mixture

Algorithm 8 NECT Recommender

```
1: procedure NECT-Train: $(\mathcal{T}; \mathbf{M}; \mathbf{x}_t = \langle x_1, \ldots, x_t \rangle; y_t = x_{t+1})$
2:     Update $\mathbf{M}^{'} = \mathbf{M}$ with $\mathbf{x}_t$ and $y_t$ // Update item co-occurrence matrix
3:     for $v_i \in \mathcal{A}(\mathbf{x}_t)$ from the root do
4:         $\alpha'_{ix} = \alpha_{ix} + 1$ // Update local expert
5:         $w'_i = \frac{w_i \mathbb{P}_i(y_t = x_{t+1}|\mathbf{x}_t)}{q_i(x)}$ // Update helper weight
6:     end for
7:     $\mathbf{x}_{t+1} = \langle \mathbf{x}_t, x_{t+1} \rangle, \mathcal{T}^{'} = \mathcal{T} \bigcup \mathcal{C}(\mathbf{x}_{t+1})$ // Expand the current tree with new suffixes
8: end procedure
9: procedure NECT-Test: $(\mathcal{T}; \mathbf{M}; \mathbf{x}_t = \langle x_1, \ldots, x_t \rangle)$
10:    Compute neighbor-guided candidate set $\mathbf{N}_{\mathbf{x}_t}$from $\mathbf{M}$
11:    for $v_i \in \mathcal{A}(\mathbf{x}_t)$ from the root do
12:        for $\hat{x} \in \mathbf{N}_{\mathbf{x}_t}$ do
13:            $\mathbb{P}_i(y_t = \hat{x}) = \frac{\alpha_{i\hat{x}} + \alpha_0}{\sum_{n \in \mathbf{N}_{\mathbf{x}_t}} \alpha_{in} + \alpha_0}$
14:            $q_i(\hat{x}) = w_i \mathbb{P}_i(y_t = \hat{x}|\mathbf{x}_t) + (1 - w_i) q_{i-1}(\hat{x})$
15:        end for
16:    end for
17:    $Top_k = \arg_k \max_{\hat{x} \in \mathbf{N}_{\mathbf{x}_t}} q_{t(\hat{x})}$ // Top-$k$ predictions
18: end procedure
```

coefficients learned through a Bayesian update.

- **NECT adapts fast to new patterns.** As more observations arrive, more contexts and nodes are added and updated in NECT. Adaptation to new items and patterns can be achieved by the non-parametric hierarchical context tree structure itself. As sessions are organized and activated by context, new items and patterns can be immediately identified through their new contexts even though they only appear a few times. Old items and patterns can still be accessed in their old contexts. Therefore, both the sample inefficiency issue on new knowledge and the knowledge retention issue on old knowledge are addressed by NECT.

As we keep adding new nodes corresponding to contexts of new sequences, the memory can be an issue in extremely large-scale recommendation applications. We can address this issue by a *garbage collection* module that keeps a record for each node and each item when it was last activated and clicked and eliminates the oldest ones when the memory usage achieves a particular upper bound. We do not encounter any memory issues through our experiments, and NECT runs smoothly within 2GB memory for different datasets without the garbage collection module. With the garbage collection module we introduced before, we managed to run our method on a local news website for over one year.

6.4 Methodology - MAN

In this section, we propose another recommendation model called Memory Augmented Neural Recommender (MAN) to address knowledge retention and data efficiency challenges for the widely used neural recommenders. Existing neural session-based recommenders typically contain two modules: an **encoder** $g_\theta(\mathbf{x}_t)$ to compute a compact *sequence representation* $\mathbf{c}_t$ for the sequence of events $\mathbf{x}_t$ until time t, and a **decoder** $f_\omega(\mathbf{c}_t)$ to compute an output distribution $P^N \in \mathbb{R}^n$ to predict the next event, where n is size of the set $\mathbf{Y}$ of all candidate items. Recurrent neural networks (RNNs) and fully-connected layers are common choices for encoder and decoder respectively. Our MAN framework is agnostic to the neural recommender; therefore, we can use many other neural architectures with an encoder-decoder structure.

Next, we present the **M**emory **A**ugmented **N**eural recommender (MAN) to augment a neural session-based recommender with a cache-like non-parametric memory to incrementally incorporate new items and preferences. We first introduce the architecture of the cache-like memory. Then we describe how to use it to generate non-parametric memory predictions and how to merge memory and neural predictions through a lightweight gating network.

6.4.1 Non-parametric Memory Structure

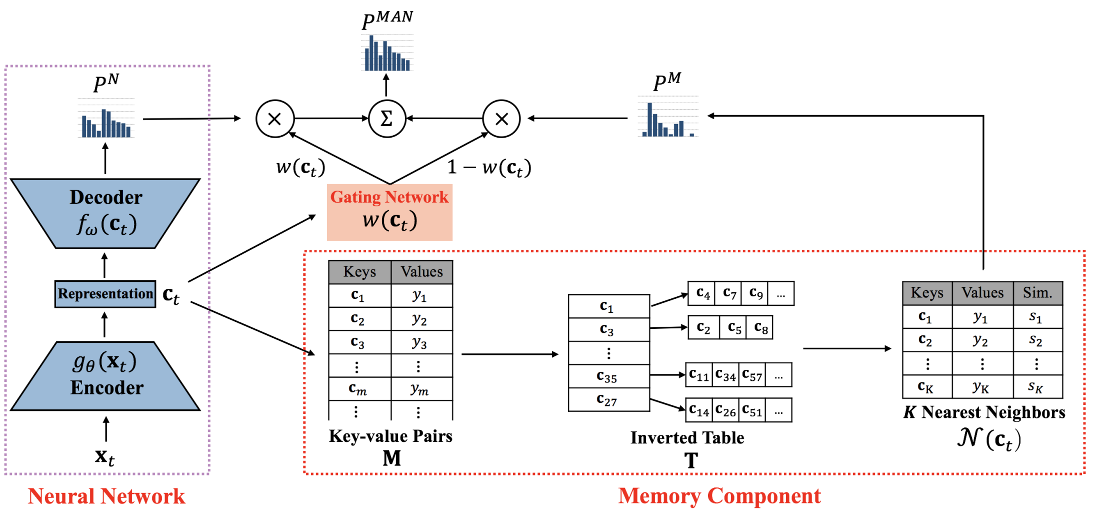
Motivated by Miller et al. (2016); Grave et al. (2017b); Sprechmann et al. (2018), we design a non-parametric memory module $\mathbf{M}$ in the form of key-value pairs, i.e., $\mathbf{M} = \{(key, value)\}$, $\mathbf{M}$ is queried by keys and returns corresponding values. We define the keys to be input sequence representations computed by the neural encoder and the values to be the corresponding label of the next event. Upon observing a training data $(\mathbf{x}_t, y_t)$, we write a new entry to $\mathbf{M}$ by:

$$\begin{cases} key \leftarrow \mathbf{c}_t = g_\theta(\mathbf{x}_t) \\ value \leftarrow y_t \end{cases} \tag{6.7}$$

To scale to a large set of observations and long histories in practical recommendation scenarios, we do not restrict the size of $\mathbf{M}$ but store all entries from previous events. Efficient retrieval and storage techniques are detailed later in Section 6.4.4. Explorations on storage issues are orthogonal to our work and will be left as future work.

6.4.2 Memory Prediction

To support efficient incremental adaptation, the memory module is *not* trained jointly with the neural recommender. Instead, it directly predicts the next event by computing



P^{MAN}
P^N
P^M
$w(\mathbf{c}_t)$
$1 - w(\mathbf{c}_t)$
Gating Network $w(\mathbf{c}_t)$
Decoder $f_\omega(\mathbf{c}_t)$
Representation $\mathbf{c}_t$
$g_\theta(\mathbf{x}_t)$ Encoder
$\mathbf{x}_t$
Keys Values
Key-value Pairs M
Inverted Table T
Keys Values Sim.
K Nearest Neighbors $\mathcal{N}(\mathbf{c}_t)$
Neural Network
Memory Component

Figure 6.3 – Computation pipeline of **M**emory **A**ugmented **N**eural recommender (MAN). Predictions from a neural network are augmented by predictions from a memory module through a gating network.

a probability distribution using entries stored in $\mathbf{M}$. For an input sequence $\mathbf{x}_t$, we first match the current sequence representation $\mathbf{c}_t = g_\theta(\mathbf{x}_t)$ against $\mathbf{M}$ to retrieve K nearest neighbors $\mathcal{N}(\mathbf{c}_t) = \{(\mathbf{c}_k, y_k, s_k)\}_{k=1}^{K}$ of $\mathbf{c}_t$. s_k is the *similarity* between the sequence representation $\mathbf{c}_k$ of the k-th neighbor and $\mathbf{c}_t$, and we use a Gaussian kernel function $s_k = exp(-||\mathbf{c}_k - \mathbf{c}_t||^2/2)$ as a measurement. Then, we use the K nearest neighbors to compute a non-parametric memory prediction $P^M \in \mathbb{R}^n$ by:

$$P^M(y_i) \propto \sum_{k=1}^{K} \delta(y_i = y_k) \cdot s_k \tag{6.8}$$

where $P^M(y_i)$ is the probability on an item $y_i \in \mathbf{Y}$, $\delta(y_i = y_k)$ is Kronecker delta which equals one when the equality holds and zero otherwise. P^M only assigns non-zero probabilities to at most K (number of neighbors) items because the probabilities assigned to items that do not appear in the nearest neighbors are zero. As a result, P^M is a very *sparse* distribution as a mixture of the labels in $\mathcal{N}(\mathbf{c}_t)$ weighted by their similarities to $\mathbf{c}_t$. To capture new preference patterns incrementally, $\mathbf{M}$ is queried and updated incrementally during the testing phase such that P^M is up-to-date.

6.4.3 Gating Network to Combine Memory and Neural Predictions

The neural prediction P^N mainly captures static old preference patterns while the non-parametric memory prediction P^M can capture infrequent and new preference patterns incrementally. To flexibly work with both scenarios, these two predictions are combined. A simple way proposed by (Grave et al., 2017a; Orhan, 2018) is linearly interpolating them with a fixed weight, and we later call this version “MAN-Shallow” in our experiments in Section 6.5.4.

To better merge these two predictions at different sequential contexts, we propose to use a lightweight **gating network** (Bakhtin et al., 2018) $w(\mathbf{c}_t)$ to learn the mixing coefficient as a function of the sequence representation $\mathbf{c}_t$. We use a lightweight fully connected neural network defined in Eq.(6.9) with a single hidden layer of 100 hidden units, *tanh* as hidden layer activation function, and a Sigmoid function at the output layer.

$$w(\mathbf{c}_t) = \sigma(\mathbf{W}_o tanh(\mathbf{W}_h \mathbf{c}_t + \mathbf{b}_h) + b_o) \tag{6.9}$$

The output of $w(\mathbf{c}_t)$ is a scalar between 0 and 1 that measures the relative importance of P^N, while the other $1 - w(\mathbf{c}_t)$ fraction is multiplied to P^M. The final prediction distribution P^{MAN} is an learning interpolation of P^M and P^N weighted by the output of $w(\mathbf{c}_t)$ computed in Eq.(6.10) .

$$P^{MAN} = w(\mathbf{c}_t)P^N + (1 - w(\mathbf{c}_t))P^M \tag{6.10}$$

The gating network is trained with cross-entropy loss using P^{MAN} as predictive distribution *after* the normal training of the neural model with both P^N and P^M are fixed. The gradients are not computed for the large number parameters of the neural model to avoid computation overhead and interference with the trained neural model. Inspired by (Bakhtin et al., 2018), it is trained using only validation data. The idea is to train it using data not seen in the training data to better predict new preferences that might appear in incremental *SR* scenarios. We randomly select 90% validation data for training and the remaining 10% for early stopping. We found that this setup achieves better performance while being much more efficient compared to using the whole training set to train the gating network.

6.4.4 Efficient Large-scale Nearest Neighbor Computation

As the number of events in practical *SR* scenarios is huge and we do not restrict the size of $\mathbf{M}$, computing nearest neighbors frequently to generate P^M can be expensive. We apply a scalable retrieval method used by (Grave et al., 2017a). To avoid exhaustive search, an *inverted file table* $\mathbf{T}$ is maintained. Keys in $\mathbf{M}$ are first clustered to a set of clusters using k-means, then all keys in $\mathbf{M}$ can be associated with one centroid. When we query $\mathbf{c}_t$ in $\mathbf{M}$, it is searched by firstly matching to a set of centroids to get the closest cluster and then the set of keys in this cluster.

The clustered memory supports efficient querying, yet it is memory consuming because each key in $\mathbf{M}$ needs to be stored. This can be greatly reduced by Product Quantization (PQ (Jégou et al., 2011)) that quantizes a vector by parts (sub-quantizers), and it does not directly store the vector but its residual, i.e., the difference between the vector and its

Algorithm 9 **M**emory **A**ugmented **N**eural Recommender

```
1: procedure MAN-TRAIN($D_{train}, D_{valid}, \mathbf{M}$)
2:     Train $g_\theta$, $f_\omega$ w.r.t. $D_{train}$
3:     for $(\mathbf{x}_t, y_t) \in D_{train}$ do
4:         Compute $\mathbf{c}_t = g_\theta(\mathbf{x}_t)$; store $(\mathbf{c}_t, y_t)$ to $\mathbf{M}$
5:     end for
6:     Build the inverted table $\mathbf{T}$ for $\mathbf{M}$
7:     Fix $g_\theta$, $f_\omega$, and train $w(\mathbf{c}_t)$ w.r.t. $D_{valid}$
8: end procedure
9: procedure MAN-TEST($D_{test}, \mathbf{M}$)
10:    for $(\mathbf{x}_t, y_t) \in D_{test}$ do
11:        Compute $\mathbf{c}_t = g_\theta(\mathbf{x}_t)$, and $P^N = f_\omega(\mathbf{c}_t)$
12:        Query $\mathbf{M}$ and $\mathbf{T}$ to retrieve $\mathcal{N}(\mathbf{c}_t)$
13:        Compute $P^M$ by Eq.(6.8), $P^{MAN}$ by Eq.(6.10)
14:        Update $\mathbf{M}$ and $\mathbf{T}$ with $(\mathbf{c}_t, y_t)$
15:    end for
16: end procedure
```

associated centroids. We use 2^8 centroids, 8 sub-quantizers per sequence representation, and 8 bits allocated per sub-quantizer, then we only need the size of 16 (quantization code + centroid id = 8 + 8) bytes per vector. Therefore, a million sequence representations can be stored with only 16 Mb memory. With an inverted table and PQ, we have a fast approximate nearest neighbor retrieval method with a low memory footprint. We use the FAISS [2] open-source library that also supports GPU acceleration for implementation. FAISS supports frequently computing nearest neighbors for up to 10 million entries.

6.4.5 Overall MAN Algorithm

The computation pipeline and algorithm of MAN presented in Figure 6.3 and Algorithm 1. Next, we describe the training and testing procedures of MAN in detail and also analyzed its computation efficiency.

Training procedure. MAN first trains the neural encoder and decoder on the training set D_{train}. Then, it computes sequence representations for all training data and stores them with corresponding labels to $\mathbf{M}$. Afterward, the clustered memory of the inverted table $\mathbf{T}$ is built with entries in $\mathbf{M}$. Lastly, The gating network is trained on the validation set D_{valid}.

Testing procedure. The sequence representation $\mathbf{c}_t$ and the neural prediction P^N is first computed. Then, $\mathbf{M}$ and $\mathbf{T}$ are queried with $\mathbf{c}_t$ to retrieve K nearest neighbors $\mathcal{N}(\mathbf{c}_t)$

[2]https://github.com/facebookresearch/faiss

	Training Data			Validation Data		Testing Data		
	Events	Sessions	Items	Events	Sessions	Events	Sessions	New Events
YOOCHOOSE	6,245,412	1,535,693	22,594	693,935	170,633	748,269	178,920	8.6%
DIGINETICA	636,506	130,994	42,294	70,723	14,555	286,254	59,240	3.3%

Table 6.1 – Statistics of two datasets. The last column indicates the percentage of testing events that involve new items not in the training set.

to compute P^M . To generate the final recommendation P^{MAN}, P^M is merged with P^N weighted by the output of the gating network. Lastly, $\mathbf{M}$ and $\mathbf{T}$ are incrementally updated with the new testing pair $(\mathbf{c}_t, y_t)$. During testing, the clustered memory in $\mathbf{T}$ is not updated for computation efficiency. Running k-means to update the clustered memory on huge data sets with large dimensions is computationally intensive. Therefore, we need to decide when and how to update the clustering algorithm, and there will be a trade-off between the performance benefits and the computation overhead. Studies on this part are left for interesting future work.

Computation efficiency analysis. During training, the additional training procedures of MAN on top of the regular neural recommender training are efficient because (i) sequence representations of training data can be obtained directly from the last regular training epoch of the neural recommender; therefore, no computation overhead is injected at this step (line 3-5); (ii) building the clustered memory, and the inverted table for entries in $\mathbf{M}$ (line 6) is also fast with the FAISS library; (iii) training the lightweight gating network using only validation split (line 7) is much more efficient than the regular training of the base neural recommender. During testing, querying the memory to retrieve nearest neighbors can be done very efficiently supported by FAISS; a forward computation through the lightweight gating network is also efficient.

6.5 Evaluation

6.5.1 Datasets and Evaluation Metrics

We used the two e-commerce datasets as mentioned in Section 5.4.1. YOOCHOOSE contains click-streams on an e-commerce site over a span of 6 months. Following previous works (Tan et al., 2016; Li et al., 2017), we use the latest 1/4 fraction of training sequences because YOOCHOOSE is quite large, and this setting generates better performances. DIGIGENTICA contains click-streams transaction data on another e-commerce site over a span of 5 months. As these two datasets are relatively static, longer testing periods (last week from YOOCHOOSE and the last four weeks from DIGIGENTICA) are assigned to the incremental test set.

Items that appear less than five times and sessions of length shorter than two or longer than 20 are filtered out. Statistics of the two datasets after pruning are summarized in Table 6.1, and the last 10% of the training data based on time is used as the validation set. Unlike previous static settings that remove items not in the training phase from test sets, our test sets for the incremental *SR* task include events on new items that appear only during the testing phase. The last column indicates the percentage of events in the test data that involve items not part of the training data.

Evaluation Metrics. As in Section 5.4.1, we use **HR@k** and **MRR@k**. HR@k is the *hit rate* that measures the fraction of top-k recommendations that contain the true item clicked next. MRR@k is the *mean reciprocal rank* of the next clicked item in top-k recommendations. The reciprocal rank of the desired item is the inverse of its rank in top-k recommended items.

6.5.2 Baseline Methods

ItemKNN: This simplest K-Nearest Neighbor method, as used in Hidasi et al. (2016), recommends items that are the most similar to the single last item in the current session. The similarity is defined by item co-occurrence statistics across all training sessions.

SKNN (Jannach and Ludewig, 2017): Instead of considering only the single latest item in ItemKNN, session-based KNN (SKNN) compares all items in the current session with the items in all other sessions.

S-SKNN (Ludewig and Jannach, 2018): A later work proposes different variations based on SKNN that assigns more weights to items appear later in a session. Different variations in (Ludewig and Jannach, 2018) show similar performance, and we include their Sequential Session-based KNN (S-SKNN).

GRU4Rec (Hidasi et al., 2016): It applies an recurrent neural network with a session-parallel mini-batch technique for optimizing variable-length sequences. Apart from the standard cross-entropy loss, they propose two ranking losses. A more recent version (Hidasi and Karatzoglou, 2018b) of GRU4Rec explores more effective ranking losses. In our experiments, we adopt the latest version of GRU4Rec implementation (Hidasi and Karatzoglou, 2018b).

NARM (Li et al., 2017): Neural Attentive Recommendation Machine (NARM) attaches an item-level attention mechanism based on the hidden representations captured by an RNN to focus on a specific part of the session. The output layer of NARM applies a bi-linear decoding scheme by directly computing a similarity score between an item embedding and the latent sequence representation, followed by a softmax layer optimized by cross-entropy loss.

η	1e-2	5e-3	1e-3	5e-4	1e-4	5e-5	0
YOOCHOOSE							
NARM	0.389	0.422	0.460	**0.463**	0.447	0.440	0.420
DIGINETICA							
NARM	0.235	0.255	0.315	0.338	**0.355**	0.350	0.324

Table 6.2 – HR@5 with different learning rate η to update NARM incrementally. NARM is fixed during testing when $\eta = 0$.

NECT (proposed in Section 6.3): It builds a non-parametric recommender based on a structure called Context Tree to model suffixes of a sequence.

MAN (proposed in Section 6.4): The method proposed in this chapter. Two versions (MAN-GRU4Rec and MAN-NARM) are tested using GRU4Rec and NARM as base neural models respectively. Unless mentioned specifically, MAN is based on NARM.

During training, the hidden layer size of GRU4Rec and NARM is set to 100, and the item embedding size of NARM is set to 50; the training batch size is set to 512; 30 epochs are trained for both models. Hyper-parameters of different models are tuned on validation splits to maximize HR@5. The number of nearest neighbors of MAN is set to 50 for YOOCHOOSE and 100 for DIGINETICA. During testing, we update different neural models and MAN incrementally using *a single gradient descent step* as every batch of 100 events are tested. Learning rates for neural models are 5e-4 and 1e-4 for YOOCHOOSE and DIGINETICA, and the learning rate to update the gating network of MAN is 1e-3. Other pure non-parametric methods (Item-KNN, (S)-SKNN, NECT) are also incrementally updated as every batch of 100 events is tested.

6.5.3 Experiment Results

In this section, we first study the effect of using different learning rates to update neural models incrementally. Then, we analyze the overall performance of different methods. Lastly, we provide in-depth analysis with an ablation study and disentangled performances with different item frequencies.

Effect of Incremental Learning Rate

Before we proceed to our main results, we first highlight that the degree of updates is important when neural models are updated incrementally. As an example, we show the results of using different learning rates to update NARM in Table 6.2. We can see that using large learning rates [3] to incrementally update NARM *degrades* performance

[3]Using multiple incremental training epochs has a similar effect as using large earning rates in one epoch.

	YOOCHOOSE				DIGINETICA			
	HR@5	MRR@5	HR@20	MRR@20	HR@5	MRR@5	HR@20	MRR@20
Item-KNN	0.205	0.114	0.403	0.127	0.112	0.042	0.186	0.056
GRU4Rec	0.359	0.216	0.582	0.228	0.191	0.114	0.382	0.135
SKNN	0.411	0.245	0.625	0.268	0.262	0.156	0.489	0.177
S-SKNN	0.416	0.247	0.628	0.272	0.279	0.170	0.497	0.185
MAN-GRU4Rec	0.447	0.269	0.657	0.293	0.331	0.203	0.545	0.226
NARM	0.463	0.280	0.682	0.303	0.358	0.221	0.566	0.242
NECT	0.464	**0.298**	0.652	**0.319**	0.377	**0.247**	0.524	**0.261**
MAN-NARM	**0.476**	0.292	**0.689**	0.314	**0.381**	0.234	**0.599**	0.258

Table 6.3 – Overall results of different models for the incremental *SR* task on two datasets. Models are ranked by HR@5, and the best method in each column is in bold.

severely, while using relatively small learning rates outperforms the version when NARM is fixed during testing ($\eta = 0$). We believe that large learning rates cause the model to overfit new patterns while catastrophically forgetting old patterns. Therefore, we contend that *incremental updates for neural models needs to be small.*

Overall Performance

The results of different methods on two datasets are summarized in Table 6.3. Several interesting empirical results can be noted:

- NECT and MAN achieve the top and comparable performance. They both outperform other baselines by notable margins. The superior performance of NECT over the state-of-the-art models is a strong signal that it is capable of learning sequential patterns incrementally. Moreover, NECT is better than MAN in terms of MRR@k where the order of items in the top-k recommendation list is considered.
- Both MAN-NARM and MAN-GRU4Rec consistently outperform their individual neural modules (NARM and GRU4Rec). This result shows that the MAN architecture effectively helps standard neural recommenders for incremental *SR* scenarios. We observed that even though NARM significantly outperforms GRU4Rec, MAN-GRU4Rec and MAN-NARM show comparable performances. We contend that the memory predictions effectively compensate for the failed predictions of GRU4Rec.

6.5.4 In-depth Analysis

Ablation Study of NECT

In this section, we study why NECT works through an ablation study. We compare NECT with three simplified versions with three different modules discarded respectively.

	YOOCHOOSE		DIGINETICA	
	HR@5	MRR@5	HR@5	MRR@5
NECT	**0.464**	**0.298**	**0.377**	**0.247**
NECT w/o Combine	0.216	0.124	0.179	0.113
NECT w/o Bayesian	0.432	0.277	0.341	0.232
NECT w/o Neighbor	0.452	0.289	0.370	0.241

Table 6.4 – Ablation study for NECT.

(i) NECT w/o Combine: the combined prediction in Eq.(6.4) is discarded and only the local expert prediction in Eq.(6.1) at the most specific suffix node is used to predict the next event. This version degenerates to a basic variable-order Markov model where the variable Markov order equals to the length of the session. **(ii) NECT w/o Bayesian:** this version computes the combined prediction in Eq.(6.2) with equal weights assigned to all active experts. Therefore, it averages all local experts, instead of using the weights computed by Eq.(6.3) and Eq.(6.5) with Bayesian inference. **(iii) NECT w/o Neighbor:** this version uses a candidate item set that includes all possible items appeared so far, instead of using the dynamically computed neighbor-guided candidate set defined in Section 6.3.4.

We can see from Table 6.4 that the performance drops very significantly if we only use the single local expert at the most specific suffix node without combining all active experts (NECT w/o Combine). It means that mixing the local predictions from the set of active experts is a critical module in NECT. Once we utilize all active experts obtained from context matching, dropping the Bayesian weight update (NECT w/o Bayesian) or discarding neighbor-guided candidate sets (NECT w/o Neighbor) hurts performance with relatively small margins. Furthermore, “NECT w/o Neighbor” requires more computations as predictions need to be computed over a much larger set of items. In conclusion, we contend that the context tree structure and the mixture of predictions from active experts with Bayesian inference play essential roles in NECT and the neighbor-guided candidate set adds extra credits to the final performance and computation efficiency.

Ablation Study of MAN

We studied in Table 6.5 the effect of two setups in MAN, i.e., the large memory size and the combining scheme with a gating network. Two simpler versions are compared. **(i) MAN-Shallow** linearly combines neural and memory predictions with a fixed scalar rather than the weight output by the gating network. The scalar weight is tuned on validation splits to be 0.7 for YOOCHOOSE and 0.8 for DIGINETICA. This simple method serves as a strong baseline in learning new vocabularies in language model tasks (Merity et al., 2017; Grave et al., 2017a). **(ii) MAN-50k/10k** use fixed-size memories

	YOOCHOOSE		DIGINETICA	
	HR@5	MRR@5	HR@5	MRR@5
MAN	**0.476**	**0.292**	**0.381**	**0.234**
MAN-Shallow	0.469	0.286	0.374	0.228
MAN-50k	0.469	0.287	0.376	0.231
MAN-10k	0.466	0.284	0.369	0.226

Table 6.5 – Ablation study for MAN. The large memory size and the gating network of MAN gain performance benefits.

Bucket #	1	2	3	4	5
YOOCHOOSE					
NECT	**0.355**	**0.286**	0.297	0.364	0.505
MAN	0.318	0.263	**0.319**	**0.378**	**0.517**
NARM	0.287	0.247	0.302	0.362	0.510
DIGINETICA					
NECT	**0.185**	**0.202**	**0.275**	0.317	0.492
MAN	0.145	0.187	0.251	**0.329**	**0.515**
NARM	0.117	0.169	0.230	0.303	0.494

Table 6.6 – Disentangled performance (HR@5) at different item frequency buckets.

that only store a limited number of recent events (50k/10k). MAN-Shallow is consistently inferior to MAN. It means the gating network makes a better decision to combine neural and memory predictions. Furthermore, the performance drops as the size of the memory decrease from unbounded (MAN) to 50k, and to 10k. It means that keeping a big memory that handles long histories achieves better recommendation performance. Despite the slight performance drop, the three simplified versions are more efficient. Therefore, they are still suitable candidates for the industry.

Disentangled Performance by Item Frequency

To further understand for what types of events that NECT and MAN most improves predictions, we studied the disentangled performance when items are bucketed into five groups ordered by their occurrence frequency in training data. Bucket 1 [4] contains the least frequent items, and bucket 5 contains the most frequent items. Bucket splitting intervals are chosen to ensure the bucket size is the same. The disentangled performances of NECT, MAN, and NARM across five buckets are reported in Table 6.6, and the results reveal that:

- Infrequent items are more challenging to predict. The performances of different

[4] Bucket 1 also contains new items in the evaluation phase that have not appeared previously.

methods have an increasing trend as item frequency increases.

- NECT outperforms both MAN and NARM on infrequent items (small bucket number), while the performance on infrequent items can be slightly worse than the other two methods. Therefore, we contend that NECT is especially good at learning new preferences on infrequent items.
- MAN consistently boosts the performance of NARM on all levels of item frequency, and the improvement margin is more significant on infrequent items. MAN performs slightly worse than NECT on infrequent items, while it is superior to NECT on frequent items.

6.6 Chapter Summary

In this chapter, we propose two initial attempts to study both knowledge retention and data efficiency in a unified framework for the session-based recommendation task. Specifically, we study how to achieve quickly learning the new items and preferences during model inference without forgetting old ones learned before. Two non-parameter methods are proposed. The first one (NECT) is based on a structure called context tree, and the second one (MAN) augments a standard neural model with a non-parametric memory prediction. We empirically show that both proposed methods outperform state-of-the-art methods through extensive experiments and analysis on two public datasets. They are especially good at predicting new or infrequent items in the evaluation phase.

We believe that this is mainly thanks to the non-parametric context tree structure in NECT and the k-Nearest Neighbor principle in MAN, which helps to memorize preferences on new coming items quickly. The two proposed approach is interesting because the memorization ability of non-parametric methods can be used to achieve long-term knowledge retention and achieve data-efficient learning on new or infrequently patterns. We hope our work could inspire more future works to consider similar principles to develop more elegant methods to achieve both knowledge retention and data efficiency for various practical applications.

In the next chapter, we study continual learning in the widely studied image classification task. We extend the widely used “Class Continual Learning” setup with a probabilistic formulation to simulate a wide range of realistic scenarios, and we highlight the importance of data augmentation in the learning process.

7 A Versatile Evaluation Protocol

7.1 Introduction

Many real-world machine learning applications require learning from data that continually arrive over time (Chen and Liu, 2018). To this end, lifelong learning (a.k.a. continual learning or continual learning), which learns from data arriving sequentially, receives increasing attention. However, different applications and tasks have different characteristics. Therefore, a standardized and versatile evaluation protocol is critical to study and compare different techniques. A widely used setting is Class Continual Learning (CCL (Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2019; Belouadah and Popescu, 2019)), where data from new classes arrive *phase* by *phase* [1]. In CCL, a set of new classes need to be learned in each phase, as depicted in Figure 7.1 (Top). The following three assumptions often exist:

1. The number of classes appearing in different phases is fixed.
2. Classes appearing in earlier phases will not appear in later phases.
3. Training samples are well-balanced across different classes in each phase.

However, these assumptions rarely hold in real-world applications. For example, in the Internet of Things (IoT) era, a deployed object recognition model needs to incrementally and periodically refine its model through data collected from different input devices (e.g., cameras, sensors) (Wen et al., 2017b). The number of different objects in an update phase is rarely balanced, and objects might *reappear* continuously. For example, in a

This chapter is based on the paper (Mi et al., 2020b) published in the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshop, 2020).

[1]'phase' is referred to as 'batch' in (Lomonaco and Maltoni, 2017; Lomonaco et al., 2019) and as 'task' in (Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2019; Belouadah and Popescu, 2019). To avoid confusion with the 'batch' in the model optimization stage or multi-'task' learning, we use 'phase' instead. This is also consistent with the notation in Chapter 4.

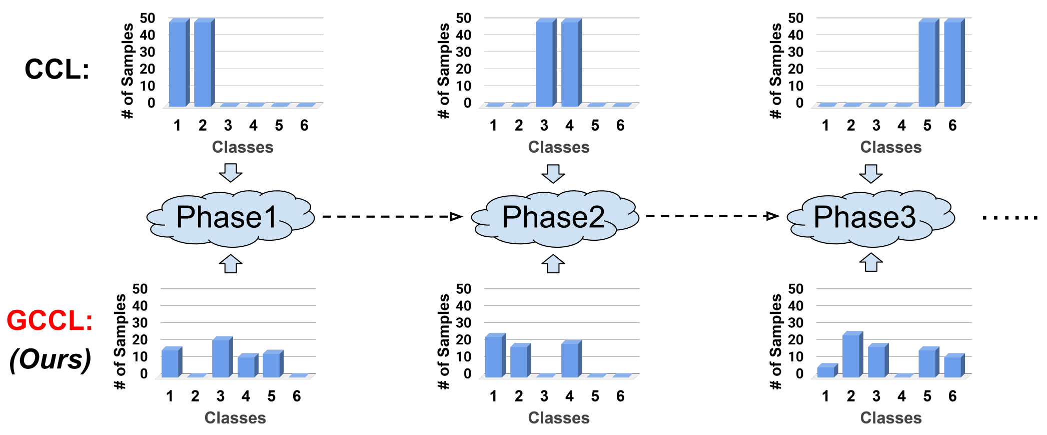


Figure 7.1 – Comparison between CCL and GCCL. **(Top)** In each phase of CCL, the model observes a fixed number of balanced classes, and classes having appeared in previous phases will not appear again. **(Bottom)** These assumptions are removed in our GCCL setting.

surveillance system at a port, a "truck" normally appear more often than a "taxi", and a "truck" will also appear in multiple update cycles. Recently, Lomonaco and Maltoni (2017); Lomonaco et al. (2019) proposed a "New Instances and Classes learning" (NIC) protocol to simulate scenarios where both new classes and new instances of each class are spread over phases for robotic applications. However, the first and third undesired assumptions described above are still not satisfied.

To ease these limitations in CCL, we propose a Generalized Class Continual Learning (GCCL) framework that allows classes to appear realistically across multiple phases, as shown in Figure 7.1 (Bottom). Precisely, we can model the incoming data sequence of each phase with three quantities: the number of classes, the appearing classes, and the sample size of each class, where each quantity is sampled from a probability distribution. We can then derive a wide range of realistic scenarios by varying these distributions.

One crucial challenge of continual learning is the catastrophic forgetting (McCloskey and Cohen, 1989; French and Chater, 2002). Many state-of-the-art CCL solutions (Lopez-Paz and Ranzato, 2017; Li and Hoiem, 2018; Hou et al., 2019; Rebuffi et al., 2017; Castro et al., 2018; Wu et al., 2019c; Hou et al., 2019; Zhao et al., 2020) are based on the common assumption that classes appear in previous phases will *not* appear in later phases. We find that these methods' superior performance cannot generalize to realistic GCCL settings as this assumption is void. There are two additional challenges in our generalized formulation, namely *data efficiency*[2], and *imbalanced class*[3], which previous CCL methods fail to overcome. To this end, we propose a simple yet effective baseline

[2]For each class, we sample its size over a distribution such that sample sizes for some classes can be small.

[3]Different classes can have very different sample sizes.

ReMix that incorporates Exemplar Replay(*ER* (Rebuffi et al., 2017; Castro et al., 2018)) to handle catastrophic forgetting and *Mixup* (Zhang et al., 2018) to address the additional data efficiency and imbalanced classes challenges. To the best of our knowledge, this is the first time *Mixup* is adopted in continual learning.

In experiments, we systematically simulate a wide range of realistic continual learning scenarios using the GCCL formulation. Our extensive analysis reveals (i) state-of-the-art methods in (Castro et al., 2018; Hou et al., 2019) perform much better than *ER* in CCL settings, however, they have similar performance as *ER* in our GCCL settings; (ii) the widely used distillation loss in CCL (Li and Hoiem, 2018; Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2019; Wu et al., 2019c) is detrimental in GCCL when *ER* is already applied and classes frequently reappear in static environments. (iii) our proposed *ReMix* consistently outperforms state-of-the-art methods by a margin of 2-6% on CIFAR-100 and 2-3% on down-sampled ImageNet. Altogether, our work is the first to establish a probabilistic formulation for realistic CCL settings, and the superior performance of *ReMix* can lead to interesting future explorations.

7.2 Formulation of Realistic Evaluation Protocol (GCCL)

Three unrealistic assumptions are present in the existing CCL protocol: (i) the number of classes appearing in each phase is fixed, (ii) the classes appearing in different phases do not overlap, and (iii) classes within a phase are balanced. To ease these limitations, we formulate a Generalized Class Continual Learning (GCCL) protocol.

First, we introduce several key concepts in use. We denote the complete set of available classes as $\mathbb{S}$ with size n. The *sample sizes* (the number of samples) of classes appearing in phase t are modeled as a random vector $\mathbf{C}_t \in \mathbb{R}^n$, where each entry $C_{t,i}$ denotes the *sample size* of class i. In the most general form, we can model any CCL scenarios by sampling $\mathbf{C}_t$ from some vector distribution. However, using a single vector distribution makes the easing of the aforementioned limitations of CCL intractable. To this end, we introduce two intermediate variables: K_t and $\mathbf{S}_t$, which denote the class number and the appearing classes at phase t respectively. Formally, the $\mathbf{C}_t$ can be formulated through three steps:

$$\begin{aligned} K_t &\sim \mathcal{D}(t) \\ \mathbf{S}_t &\sim \mathcal{R}(\mathbf{W}_t^1, K_t) \\ \mathbf{C}_t &\sim \mathcal{M}(\mathbf{W}_t^2, \mathbf{S}_t)\,, \end{aligned} \tag{7.1}$$

where K_t, $\mathbf{S}_t$, and $\mathbf{C}_t$ are sampled from three distributions $\mathcal{D}$, $\mathcal{R}$, and $\mathcal{M}$. $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$ are two weight vectors determining the appearing probability and sample size of classes in phase t.

The above three-step formulation can simulate a wide range of continual learning scenarios with the following three essential *properties* in many real-world applications.

1. A different number of classes in different phases. The number of classes K_t appearing in phase t follows a phase-dependent discrete distribution $\mathcal{D}(t)$. Setting $\mathcal{D}(t)$ to a constant results in a fixed K_t across different phases as in traditional CCL settings. Therefore, by choosing a $\mathcal{D}(t)$ satisfying for two different phases t and t'

$$P(K_t \neq K_{t'}) > 0\,, \tag{7.2}$$

we ease the first limitation of CCL to allow *a variable number of classes in different phases.*

2. Overlapping/Reappearing classes in different phases. Classes appearing in phase t are modeled as a random vector $\mathbf{S}_t \in \mathbb{R}^n$. $\mathbf{S}_t$ is a binary indicator vector with ones corresponding to classes in phase t. It is sampled from a distribution $\mathcal{R}(\mathbf{W}_t^1, K_t)$ which depends on the class number K_t and a *class appearance weight vector* $\mathbf{W}_t^1 \in \mathbb{R}^n$. $\mathbf{W}_t^1$ represents the appearing probability of each class in phase t. Classes with high appearing probability are more likely to appear in different phases. In contrast, classes with low appearing probability might disappear for a long sequence of phases so that they tend to be forgotten. Different from previous CCL protocols, $\mathcal{R}$ does not necessarily have disjoint supports across phases such that the following property is satisfied: for two different phases t and t',

$$P(\mathbf{S}_t \odot \mathbf{S}_{t'} \neq \mathbf{0}) > 0\,, \tag{7.3}$$

where $\odot$ denotes element-wise multiplication of two vectors. Therefore, the second limitation of CCL is lifted to allow *classes to reappear in later phases.*

3. Different sample sizes for classes in a phase. The last step is to determine each appearing class's sample size in the current phase, which is encoded as a random vector $\mathbf{C}_t$. $\mathbf{C}_t$ follows a distribution $\mathcal{M}(\mathbf{W}_t^2, \mathbf{S}_t)$, which depends on the appearing classes $\mathbf{S}_t$ and a *class sample size weight vector* $\mathbf{W}_t^2 \in \mathbb{R}^n$. $\mathbf{W}_t^2$ determines the sample sizes of different classes appearing in phase t. Varying $\mathbf{W}_t^2$ can model different degrees of imbalanced classes issue within a phase, and this formulation fulfills the following property: for any two different class i and j,

$$P(C_{t,i} \neq C_{t,j} | C_{t,i} \neq 0, C_{t,j} \neq 0) > 0\,. \tag{7.4}$$

Therefore, the last limitation of CCL is released in GCCL to allow *sample sizes of classes in a phase to be different.* We stress that $\mathbf{W}_t^2$ is intrinsically different from $\mathbf{W}_t^1$. A class might frequently appear among different phases in certain scenarios, but it only appears with a small quantity per phase. Therefore, such classes have large weights in $\mathbf{W}_t^1$ but small weights in $\mathbf{W}_t^2$, and we study such special situations in Section 7.4.3.

Remark 1. In GCCL, the reappearance of previous classes alleviates the catastrophic forgetting challenge. However, it still exists because some classes might disappear for a long sequence of phases.

Remark 2. The above GCCL formulation requires to tackle two more challenges other than the catastrophic forgetting challenge:

- **Data efficiency**: GCCL allows sample sizes of appearing classes to be much smaller than that in CCL to reflect potential data scarcity of some classes. Therefore, data efficiency needs to be improved to learn from a limited amount of data.
- **Imbalanced classes**: GCCL allows classes to be imbalanced within a phase or across different phases; therefore, the model needs to handle imbalanced classes.

Remark 3. The GCCL formulation covers a wide range of continual learning scenarios. For example, traditional CCL protocols (Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2019; Belouadah and Popescu, 2019), as well as the recent "New Instances and Classes learning" (NIC (Lomonaco and Maltoni, 2017; Lomonaco et al., 2019)) protocol, are special instances of our framework.

7.3 Methodology – *ReMix*

To tackle the aforementioned challenges in GCCL, we propose a simple yet effective method called *ReMix*, which combines Exemplar Replay and *Mixup*.

Exemplar Replay (*ER*) based on Herding. *ER* methods (Rebuffi et al., 2017; Castro et al., 2018; Hou et al., 2019; Wu et al., 2019c) have shown great success in CCL to mitigate catastrophic forgetting. *ER* stores a couple of exemplars per class for all experienced classes until the current phase. All exemplars in previous phases are combined with data in the current phase to update the model. We adopt the *Herding* (Welling, 2009; Rebuffi et al., 2017) technique to select exemplars. *Herding* chooses exemplars of a class that best approximate the *average feature vector* over all training examples of this class till the current phase. Our later experiments in Section 7.4.5 justify the benefits of *Herding* compared to other exemplar selection schemes.

***Mixup*.** *ER* addresses catastrophic forgetting, but the issues of data efficiency and imbalanced classes remain unsolved. To address these two challenges, we introduce the data augmentation technique *Mixup* (Zhang et al., 2018) as a complementary component to *ER*.

The idea of *Mixup* is simple: it creates *virtual training samples* through a linear interpolation between raw training samples in order to learn smooth decision boundaries between all classes. Formally, a virtual training sample $(\tilde{x}, \tilde{y})$ is generated through a convex combination between a pair of raw samples (x_i, y_i) and (x_j, y_j) by:

$$\tilde{x} = \lambda x_i + (1 - \lambda)x_j, \tilde{y} = \lambda y_i + (1 - \lambda)y_j,$$

where x_i and x_j are features of two input; y_i and y_j are the corresponding labels; $\lambda \sim Beta(\alpha, \alpha)$ with hyperparameter $\alpha \in (0, \infty)$.

Mixup has shown great success in different image classification (He et al., 2019b; Thulasidasan et al., 2019) and object detection (Zhang et al., 2019; Yun et al., 2019) tasks. Nevertheless, it has not been studied for continual learning scenarios.

ReMix* formulation.** We propose to use *Exemplar* ***Re*play* together with ***Mix****up*, referred to as *ReMix*. As in *ER*, exemplars of different classes are selected by the *Herding* technique, and exemplars in earlier phases are replayed in the current phase. In each incremental training phase, *Mixup* is applied to training mini-batches containing both samples in the current phase and exemplars in previous phases replayed by *ER*. In Section 7.4.5, we empirically demonstrate the benefits of (i) using *Mixup* together with *ER*, and (i) interpolating exemplars with samples in the current phase.

Three desirable properties of *ReMix* are analyzed below:

1. As a data augmentation method, the limited number of exemplars are augmented to further mitigate catastrophic forgetting. Similarly, classes with insufficient samples are also augmented to improve data efficiency.
2. The regularization effect of *ReMix* helps to deal with imbalanced classes. It prevents the model from overfitting dominant classes by smoothing decision boundaries among all classes.
3. *ReMix* can be easily applied to different scenarios. It requires no restrictive assumptions on the incoming data. Also, it introduces minimal computation overhead without extra training data nor model parameters.

7.4 Evaluation

In this section, we first describe a wide range of state-of-the-art methods evaluated in our experiments. In Section 7.4.2, we introduce various realistic GCCL setups in our experiments that are not yet explored. Then, we present results of different GCCL setups on CIFAR-100 and ImageNet in Section 7.4.3 and Section 7.4.4 respectively. Lastly, we conduct a case study with detailed analysis on *ReMix* in Section 7.4.5.

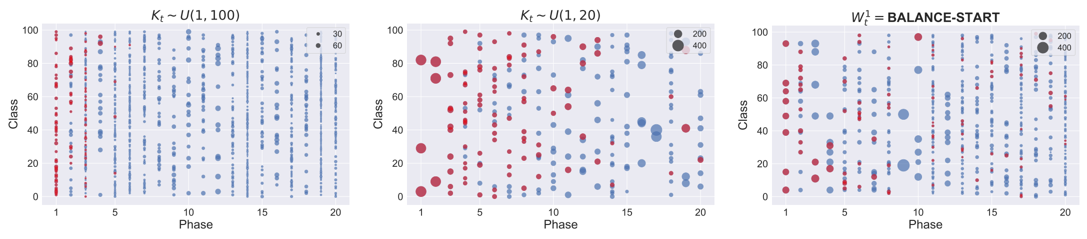


Figure 7.2 – Visualizing appearing classes at different phases on CIFAR-100. **Left**: $K_t \sim \mathcal{U}(1,100)$. **Middle**: $K_t \sim \mathcal{U}(1,20)$; **Right**: $\mathbf{W}_t^1$ =BALANCE-START. Red circles represent new classes and blue circles represent old ones. The circle size is proportional to the sample size of the class. Visualizations for other test scenarios are not included because they are straightforward.

7.4.1 Baseline Methods

We evaluate a wide range of representative baselines:

- ***Finetune***: The model is updated with data in the current phase without extra techniques nor exemplars.
- ***LwF*** (Li and Hoiem, 2018): A knowledge distillation loss is applied on top of *Finetune* to prevent prediction logit changes on past classes not in the current phase.
- ***ER*** (Chaudhry et al., 2019; Rebuffi et al., 2017): It updates the model with data in the current phase and exemplars in previous phases selected by *Herding*. Our default setting stores 20 exemplars per class for CIFAR-100 and 50 exemplars for ImageNet. Other exemplar sizes are tested in Section 7.4.5.
- ***iCaRL*** (Rebuffi et al., 2017): Based on the *ER* method, *iCaRL* applies a non-parametric nearest-mean-of-exemplars rule for classification.
- ***GEM*** (Lopez-Paz and Ranzato, 2017): For each update, current gradients are projected to a feasible region formed by exemplar gradients. A ring buffer stores an equivalent number of exemplars as in *ER* for each phase.
- ***CN*** (Hou et al., 2019): **C**osine **N**ormalization is applied to both inputs and weights of the output layer in *ER* to deal with imbalanced classes.
- ***BF*** (Castro et al., 2018): After the regular training procedures as in *ER*, *BF* finetunes the output layer with balanced exemplars to address imbalanced classes issue.
- ***Full***: For each phase, the model is trained with data from the current phase and *all* previous phases. This is a common performance "upper bound" in CCL.
- ***ReMix* (Proposed)**: The method introduced in section 7.3.

7.4.2 Various Realistic GCCL Experiment Setups

In our experiments, we use two image classification datasets, CIFAR-100 (Krizhevsky et al., 2009) and ImageNet (Russakovsky et al., 2015) [4]. We test 20 incremental training phases, and the phase size is set to 1,000 on CIFAR-100, and 5,000 on ImageNet [5]. We use a 32-layer ResNet (He et al., 2016) for all experiments. The network is trained by stochastic gradient descent with a mini-batch size 100. For both datasets, we use standard data augmentation (random crop and horizontal flip). All models are trained using the same training protocol. On CIFAR-100, each phase is trained with 100 epochs. The learning rate starts from 0.1 and is divided by 10 after 60 and 80 epochs; weight decay is 5e-4, and momentum is 0.9. On ImageNet, each phase is trained with 60 epochs. The learning rate also starts from 0.1 and is divided by 10 after 36 and 48 epochs; weight decay is 1e-4, and momentum is 0.9. Models are evaluated on the balanced test set consisting of all classes that have appeared so far, and we report *Average TOP-1 accuracy* over all phases. Three sets of experiments are designed to study the effects of the three key factors in our GCCL formulation introduced in Section 7.2.

1. Varying class numbers. We choose $\mathcal{D}(t)$ as fixed uniform distributions $\mathcal{U}(1, u)$ to sample the number of classes K_t and evaluate different supports $(\mathcal{U}(1, 20), \mathcal{U}(1, 50), \mathcal{U}(1, 100))$. When u is large, the number of classes in a phase tends to be large. While testing different u's, we set $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$ to be uniform distribution (UNIFORM) over all all classes in $\mathbb{S}$. In Figure 7.2 (left and middle), we visualize two cases (u set as 100 and 20 accordingly) of appearing classes at different phases.

2. Varying appearing classes. We choose "sampling *without* replacement" as the realization of the distribution $\mathcal{R}$. Two types of $\mathbf{W}_t^1$ other than UNIFORM are evaluated:

- **BALANCE-START**: $\mathbf{W}_t^1$ is a phase-dependent distribution based on which new class appearance is distributed evenly over all phases. Specifically, the first appearing time of different classes is sampled uniformly across the total number of phases. Next, we encode this information by setting the weights of classes that do not appear before phase t to 0 in $\mathbf{W}_t^1$. With the phase-dependent $\mathbf{W}_t^1$ defined in this way, we can sample K_t from $\mathcal{U}(1, u)$, with u being a phase-dependent number computed from the number of nonzero entries in $\mathbf{W}_t^1$, to allow the class number of different phases to be a variable.
- **LONGTAIL**: $\mathbf{W}_t^1$ is a fixed long-tailed distribution, where the weight $w_{t,i}$ for class i is generated by an exponential function $w_{t,i} = \mu^i$ (Cui et al., 2019). Different

[4] we use the first 200 classes of ImageNet and down-sampled images to 32 × 32 for computation feasibility.

[5] We focus on varying different realistic incoming data distributions. More evaluations on a different number of phases and phase sizes are left for future work.

	Varying K_t		
	$\mathcal{U}(1,20)$	$\mathcal{U}(1,50)$	$\mathcal{U}(1,100)$
Full	54.31 (2.82)	46.25 (2.11)	44.32 (0.90)
ReMix	**41.72** (2.01)	**36.75** (1.05)	**37.80** (0.55)
CN	39.63 (2.31)	34.05 (1.59)	32.89 (2.09)
BF	40.71 (2.41)	34.85 (1.67)	32.13 (1.13)
ER	38.54 (1.91)	34.07 (1.89)	31.73 (1.21)
iCaRL	40.89 (1.95)	34.48 (1.77)	31.43 (1.30)
GEM	19.52 (1.35)	25.36 (1.57)	24.86 (1.36)
LwF	14.58 (1.75)	15.43 (1.46)	17.15 (0.67)
Finetune	14.33 (2.04)	13.24 (2.02)	14.61 (0.91)

Table 7.1 – Average Top-1 accuracy (%) over all phases on *CIFAR-100* with Varying K_t. Standard deviation over 5 runs with different random seeds is reported in parenthesis.

μ's correspond to different degrees of the imbalanced classes issue. We set it to have the largest weight 5 times larger than the smallest. We set $K_t \sim \mathcal{U}(1,100)$ and $\mathbf{W}_t^2 = \text{UNIFORM}$.

3. Varying sample sizes. We choose a multinomial distribution as a realization of the distribution $\mathcal{M}$. $\mathbf{W}_t^2$ can be various distributions, other than UNIFORM, to reflect different real-world scenarios:

- **TASK-VARIED**: $\mathbf{W}_t^2$ varies across different phases by adding a Gaussian noise (0 mean and 20% of uniform class weight as standard deviation) on top of UNIFORM.
- **LONGTAIL**: $\mathbf{W}_t^2$ takes the same fixed long-tail distribution as introduced for $\mathbf{W}_t^1$.

While testing different $\mathbf{W}_t^2$, we set $K_t \sim \mathcal{U}(1,100)$ and $\mathbf{W}_t^1 = \text{UNIFORM}$.

7.4.3 Results on CIFAR-100

Table 7.1 and 7.2 summarize the overall performance of different methods in different GCCL setups averaged over all 20 phases. We first analyze performance patterns of different setups, then we compare different methods and empirically show that *ReMix* outperforms state-of-the-art baselines by a significant margin.

Comparison among GCCL setups

- Setups that require high data efficiency are more challenging. In Table 7.1, as

	Varying $\mathbf{W}_t^1$		Varying $\mathbf{W}_t^2$	
	BALANCE-START	**LONGTAIL**	**TASK-VARIED**	**LONGTAIL**
Full	58.21 (1.90)	43.55 (3.01)	42.99 (1.77)	41.76 (1.04)
ReMix	**47.76** (1.41)	**35.95** (2.33)	**36.27** (1.22)	**35.85** (0.63)
CN	46.49 (2.41)	31.53 (3.14)	31.31 (1.56)	29.87 (1.35)
BF	45.90 (1.63)	30.83 (3.01)	30.14 (1.43)	29.82 (1.08)
ER	45.62 (2.21)	30.69 (3.25)	30.93 (1.96)	30.63 (0.76)
iCaRL	44.11 (2.07)	30.83 (2.82)	30.36 (2.02)	29.40 (0.85)
GEM	34.42 (3.26)	24.00 (2.21)	25.12 (2.01)	24.80 (0.78)
LwF	26.33 (2.08)	15.74 (3.64)	16.21 (2.24)	15.05 (0.89)
Finetune	25.49 (2.66)	14.76 (3.96)	16.04 (2.38)	14.59 (0.87)

Table 7.2 – Average Top-1 accuracy (%) over all phases on *CIFAR-100* with Varying $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$. Standard deviation over 5 runs with different random seeds is reported in parenthesis.

u in $K_t \sim \mathcal{U}(1,u)$ increases from 20 to 50 and 100, the sample size per class is reduced due to the fixed phase size and increased appearing classes per phase. The performance of most methods, except *GEM* and *LwF*, shows a decreasing trend due to the increased data efficiency challenge. Similarly, all methods perform well when $\mathbf{W}_t^1$=BALANCE-START as the data efficiency challenge is less significant.

- Complex class arriving patterns (e.g. TASK-VARIED and LONGTAIL) modeled by $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$ are more challenging than the uniform assumption in previous CCL settings. Most methods perform worse when $\mathbf{W}_t^1 =$ LONGTAIL or $\mathbf{W}_t^2 =$ LONGTAIL/TASK-VARIED, compared to the setup when they are UNIFORM (column 4).

Comparison among baseline methods

- Exemplar Replay methods (*iCaRL*, *ER*, *CN*, *BF*) perform better than regularization methods (*GEM*, *LwF*) with notable margins. *iCaRL* often out-performs *ER* in CCL setups (Hou et al., 2019; Rebuffi et al., 2017), however, their performance gap is negligible in various GCCL setups. *CN* and *BF* which rectify the bias of the output layer of *ER* significantly outperforms *ER* in CCL setups (Hou et al., 2019; Wu et al., 2019c). Nevertheless, they only yield marginal improvements over *ER* in realistic GCCL setups.
- Classical regularization methods (*LwF*, *GEM*) do not achieve promising performance. We can see that *LwF* is only slightly better than *Finetune*. *GEM* is notably better than *LwF*; however, it is still inferior to other *ER* based methods. This is because regularization methods restrain the model from learning reappearing

γ	0	0.1	0.25	0.5	1	2
UNIFORM	31.73	31.17	30.76	29.77	28.83	27.78
TASK-VARIED	30.93	30.23	29.85	28.79	27.91	27.02
LONGTAIL	30.63	29.93	29.15	28.12	27.06	26.23

Table 7.3 – The distillation loss is *detrimental* in GCCL when *ER* is already applied. Average Top-1 accuracy (%) on variations of $\mathbf{W}_t^2$ on CIFAR-100. Different weights (γ) of distillation loss are added to the regular cross-entropy loss of *ER* ($\gamma = 0$).

	$\mathbf{W_t^1}, \mathbf{W_t^2}$ SAME-LONGTAIL	$\mathbf{W_t^1}, \mathbf{W_t^2}$ REVERSE-LONGTAIL
Full	39.73 (0.74)	41.41 (1.63)
ReMix	**33.83** (0.45)	**34.12** (1.06)
CN	30.96 (1.02)	30.79 (1.27)
ER	29.31 (1.11)	30.06 (1.71)
GEM	22.61 (1.83)	23.04 (2.25)
Finetune	14.53 (1.69)	13.29 (2.30)

Table 7.4 – Average Top-1 accuracy (%) for two other GCCL setups with varying $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$. We set $K_t \sim \mathcal{U}(1, 100)$. Standard deviation over 5 runs with different random seeds is reported in parenthesis.

classes in GCCL. We further note in Table 7.3 that the widely used distillation loss (Li and Hoiem, 2018; Rebuffi et al., 2017; Hou et al., 2019; Wu et al., 2019c) is *detrimental* when *ER* is already applied. The distillation loss in each phase is applied to classes not appear in the current phase. We can observe that high weights on the distillation loss lower the original *ER* performance ($\gamma = 0$), which means that the distillation loss is not helpful when classes frequently reappear.

- *ReMix* outperforms other methods by notable margins (2-6% over the closest competitor) in all new settings. Specifically, our method shows multiple advantages: (i) better data efficiency (e.g., 5% margin under $K_t \sim \mathcal{U}(1, 100)$); (ii) better ability to overcome catastrophic forgetting (e.g., 4% margin under $\mathbf{W}_t^1$ = LONGTAIL); and (iii) more robust to imbalanced classes issue (e.g., 5% margin under $\mathbf{W}_t^2$ = LONGTAIL). This result demonstrates the effectiveness of *ReMix* in different realistic continual learning scenarios.

Other GCCL Setups

In this experiment, we evaluate two other special scenarios to set $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$. In the first scenario (SAME-LONGTAIL), $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$ are the same long-tailed distribution as described in Section 7.4.2. Classes that appear more often also appear by larger quantity

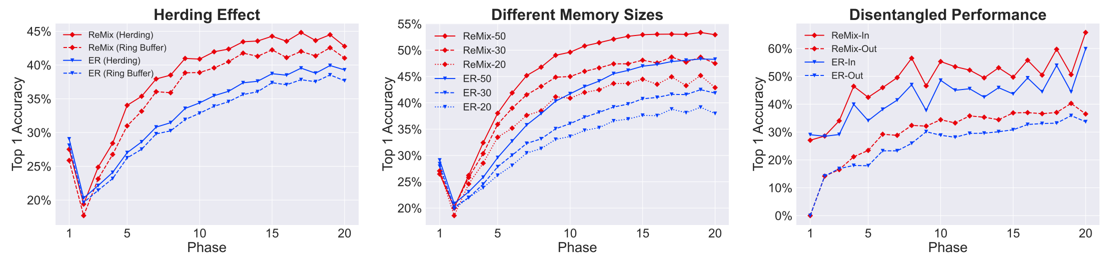


Figure 7.3 – Detailed analysis of *ReMix* with different settings on CIFAR-100. **(Left)** Herding v.s. Ring Buffer to select exemplars for *ReMix* and *ER*. **(Middle)** Performance of *ReMix* and *ER* with varying exemplar sizes. **(Right)** Disentangled performance comparing *ReMix* with *ER* regarding classes in the current phase or not.

in different phases. In the second scenario (REVERSE-LONGTAIL), $\mathbf{W}_t^1$ and $\mathbf{W}_t^2$ are set as long-tailed distributions with reversed head. To be specific, the weight of class i is set as μ^i in $\mathbf{W}_t^1$ and $\mu^{(100-i)}$ in $\mathbf{W}_t^2$. Therefore, classes have a large weight in $\mathbf{W}_t^1$ have a small weight in $\mathbf{W}_t^2$, and vice versa. In this case, a class might frequently appear among different phases, but it tends to appear with a small quantity in each phase.

Results of several representative methods in these two GCCL setups are presented in Table 7.4 with $K_t \sim \mathcal{U}(1, 100)$. We can note very similar performance patterns as the in Table 7.1 and 7.2. *CN* and *ER* are still better than *GEM*, while *ReMix* is still superior to both of them.

7.4.4 Results on ImageNet

We present ImageNet results in Table 7.5 of several representative methods when $K_t \sim \mathcal{U}(1, 100)$, $\mathbf{W}_t^1 = \text{UNIFORM}$, and $\mathbf{W}_t^2 \in \{\text{UNIFORM, TASK-VARIED, LONGTAIL}\}$. Similar performance patterns, as in CIFAR-100, can be observed. The improvement of *ER* over *Finetune* shows the benefit of using exemplars. *ReMix* still outperforms *ER* and other methods by substantial margins. Therefore, we contend the superior performance of *ReMix* and the analyses we conduct for CIFAR-100 can generalize to more challenging and large-scale scenarios.

7.4.5 In-depth analysis on *ReMix*

In this section, we analyze the success of *ReMix* in detail. As a case study, we limit our discussion on CIFAR-100 with $K_t \sim \mathcal{U}(1, 100)$, $\mathbf{W}_t^1$ = UNIFORM, $\mathbf{W}_t^2$ = TASK-VARIED.

	UNIFORM	TASK-VARIED	LONGTAIL
Full	37.79 (0.79)	37.57 (1.30)	36.25 (1.23)
ReMix	**25.39** (1.36)	**25.17** (1.45)	**24.58** (1.15)
CN	23.13 (1.01)	22.76 (1.51)	22.39 (1.12)
ER	22.92 (1.12)	22.27 (1.41)	21.97 (1.43)
GEM	14.81 (1.24)	14.95 (1.61)	14.38 (1.39)
Finetune	11.62 (1.17)	11.55 (1.21)	11.02 (1.28)

Table 7.5 – Average Top-1 accuracy (%) on ImageNet with varying $\mathbf{W}_t^2$. We set $K_t \sim \mathcal{U}(1, 100)$ and $\mathbf{W}_t^1 =$UNIFORM. Standard deviation over 5 runs with different random seeds is reported in parenthesis.

ReMix	*ReMix-v1*	*ReMix-v2*	*Mixup*	*ER*
36.27	34.52	32.39	15.93	30.93

Table 7.6 – Ablation study for *ReMix*.

Herding is effective. Other than *Herding*, we test another exemplar management scheme using a *Ring Buffer* to select the most recent samples as exemplars for each class. The exemplar sizes of *Ring Buffer* and *Herding* are the same. The performance comparison of these two schemes is presented in Figure 7.3 (Left). We can see that *Herding* is consistently better than *Ring Buffer* across all incremental training phases for both *ReMix* and *ER*.

Different memory sizes. In Figure 7.3 (Middle), we vary the number (20, 30, 50) of exemplars per class selected by *Herding*. The increased exemplar size improves both *ER* and *ReMix*. Nevertheless, *ReMix* is consistently superior to *ER* with different exemplar sizes.

Disentangled performance. In Figure 7.3 (Right), we show the performance on classes in the current phase (*in-phase* classes) and not in the current phase (*out-phase* classes) separately. Improvements in *out-phase* classes demonstrate that *ReMix* alleviates catastrophic forgetting issue on classes not in the current phase. Improvements in *in-phase* classes show that *ReMix* improves data efficiency to learn classes in the current phase quickly. *Altogether, this result shows that* ReMix *helps to better retain previous knowledge and to improve data efficiency.*

Ablation study for *ReMix*. In Table 7.6, three variants of *ReMix* are evaluated to show the benefits of combining *ER* with *Mixup*. The fact that *Mixup* (w/o exemplars) alone fails badly shows that exemplars are crucial for *ReMix*. In *ReMix-v1*, *Mixup* is only performed among exemplars, while data in the current phase are raw. In *ReMix-v2*, *Mixup* is only performed on data in the current phase, while exemplars are raw. Although *ReMix-v1* and *ReMix-v2* outperform *ER*, they are both inferior to *ReMix*, which justifies the importance of interpolating exemplars with samples in the current phase.

7.5 Chapter Summary

This chapter revisits the oversimplified and unrealistic setup in the current class continual learning research and proposes a Generalized Class Continual Learning (GCCL) framework. The probabilistic nature of GCCL allows it to simulate a wide range of realistic continual learning scenarios to serve as a versatile benchmark. We identify new challenges in GCCL and reveal the shortage of previous methods. To this end, we propose a simple yet efficient method, *ReMix*, that combines *Exemplar Replay* and *Mixup*. We simulate a wide range of realistic scenarios on CIFAR-100 and down-sampled ImageNet, and our extensive empirical evaluations demonstrate that *ReMix* consistently outperforms all previous methods. It improves both knowledge retention and data efficiency. We hope our proposed GCCL formulation could serve as a more generalized evaluation protocol to motivate more research ideas towards realistic lifelong learning.

8 Conclusion

8.1 Summary

Machine learning has been instrumental for both data analysis and artificial intelligence. It brings enormous benefits and convenience to computer science, natural science, life science, social sciences, engineering, and beyond. More tangibly, machine learning techniques gradually changes and improves our individual life and society. Massive practical applications and services are enabled and improved by a wide range of "hidden" machine learning models.

We have witnessed the great success of machine learning in recent years with the prosperity of different deep learning techniques and exponentially increased computation power and data scale. For example, the image classification performance is boosted by large-scale ImageNet dataset (Russakovsky et al., 2015) and deep neural network architectures; AlphaGo (Silver et al., 2017) achieves extraordinary performance at the game of Go by optimizing over a huge number of Go scores; large-scale pre-trained language models demonstrate excellent natural language understanding capability by training a big neural model on a massive corpus. Nevertheless, most disruptive innovations and progress fall into the category of a classic machine learning paradigm. That is, the model needs to be trained on a big static dataset and evaluated w.r.t. new data of the same task.

However, in many real-life applications, machine learning models need to continuously learn new tasks, domains, distributions, etc. The classic machine learning paradigm built on top of the independent and identical distribution assumption cannot do the job well. Therefore, we study a lifelong machine learning paradigm towards this goal, which describes a continuous learning process with the ability to retain and accumulate knowledge and use it to facilitate quick future learning as humans often do. This thesis focuses on two major challenges of lifelong learning. The first **data efficiency** challenge is to learn new knowledge with a small number of observations. The second **knowledge retention** challenge is to prevent a machine learning system from forgetting the old knowledge it has previously learned.

In Part I of this thesis, we study how to improve data efficiency for task-oriented dialog systems. As the annotation cost in task-oriented dialog systems is very high, training a model with a small number of annotations is a practically important topic. The first proposed approach (Chapter 2) is based on Meta-learning, aiming to learn a better model parameter initialization that can quickly reach a good parameter region of new domains or tasks with a small number of labeled data. More specifically, we propose a Meta-learning approach base on MAML (Finn et al., 2017) for the natural language generation module. Our experiments and analysis reveal that the proposed method indeed learns new domains faster and better with a small number of labeled data. The second proposal (Chapter 3) takes a semi-supervised self-training approach to iteratively train a better model using the sufficient unlabeled data when only a limited number of labeled data are available. We propose a self-training method to gradually train a stronger model by iteratively labeling the most confident unlabeled data and a new text augmentation technique called GradAug making use of the masked language model of BERT (Devlin et al., 2019a). We conduct extensive experiments on four common downstream tasks in task-oriented dialog systems, including intent classification, dialog state tracking, dialog act prediction, and response selection. Empirical results demonstrate that the proposed self-training technique effectively improve data efficiency, and it consistently improves state-of-the-art pre-trained models.

In Part II, we study the knowledge retention challenge. A continual learning setup is formulated and studied in different applications to combat the detrimental catastrophic forgetting issue when neural networks learn new knowledge sequentially. Two chapters in Part II consider the continual learning setup for the domain of task-oriented dialog systems and recommendation systems, respectively. Through extensive evaluation and analysis, we summarize two findings regarding mitigating the catastrophic forgetting issue: (1) it is simple but effective to replay a small number of representative exemplars, i.e., storing representative historical data and replaying them to the model while learning new tasks; (2) it is helpful to put an additional and dynamic constraint on top of "exemplar reply" for not forgetting previously learned knowledge while learning new tasks.

In Part III, we attempt to achieve both data efficiency and knowledge retention in a unified framework. Chapter 6 focuses on recommendation systems which naturally have both concerns. That is, a recommendation system in dynamic environments needs to capture new items and preferences with limited observations, and it also needs to maintain its knowledge of old items and preferences. We propose two methods using two non-parametric methods with context tree and k-Nearest Neighbor, respectively. We demonstrate that the two proposed non-parametric memory modules help to retain long-term knowledge. More importantly, the proposed non-parametric prediction computed on top of them helps to capture new knowledge in a data-efficient manner. This finding is also aligned with the recent results on the memorization ability of k-Nearest Neighbor

(Cohen et al., 2018; Khandelwal et al., 2020). To provide a versatile and generalized evaluation protocol to study different techniques, Chapter 7 proposes a probabilistic formulation to simulate a wide range of realistic lifelong learning scenarios with knowledge retention and data efficiency concerns for the image classification task. Through extensive empirical evaluation, we also show the benefit of data augmentation using Mixup to improve knowledge retention and data efficiency.

8.2 Future Directions

With the ever-increasing real-world use of machine learning, lifelong learning gradually becomes vital for devising robust and scalable models in different applications. So far, the lifelong learning ability of the machine learning system still largely lacks human-level capability, and lifelong learning studies are still not mature. Hence, there are a number of potential directions for future work.

With the recent advance of large-scale models pre-trained on massive general knowledge, such as texts or images, the few-shot learning ability is greatly improved (Brown et al., 2020; Su et al., 2020). In Chapter 3, we consider using unlabeled data to further improve the pre-trained models. However, some applications might not have sufficient unlabeled data. Hence, how to improve the data efficiency in this case on top of strong pre-trained models is another challenge. For example, how to design a better fine-tune procedure for learning a downstream task with better data efficiency? As the pre-trained models are often very large and general, we contend that it is interesting to explore how to selectively fuse or distill a subset of the large number of important parameters for a downstream task to improve data efficiency.

Another crucial topic for lifelong machine learning is constructing and utilizing the knowledge base (KB). As we mentioned before, a fundamental challenge for lifelong machine learning is retaining knowledge to facilitate future learning on new tasks. Instead of retaining knowledge within the representations and parameters of a machine learning model, knowledge can be explicitly expressed and retained as domain-specific or task-specific KB entries or graphs in many real-world applications. Such systems often have a large and increasing number of entries in their KB. How to reason over a large KB is a popular research area in the machine learning community. However, it remains to explore how to better accumulate and maintain the knowledge in the KB and how to use the accumulated past knowledge to help future learning in a data-efficient manner. The research about KB is not conducted in this thesis, and some related explorations can be seen at Chen and Liu (2018) for a survey.

In Chapter 6, we attempt to deal with knowledge retention and data efficiency challenges in a unified framework by using non-parametric methods. More sophisticated approaches can be explored as future works. We hope that more fundamental learning rules can be proposed inspired by biology or neuroscience concepts. For example, neuro synaptic plasticity is an essential feature of our brain. It yields physical changes in the neural structure and allows us to learn, remember, and adapt to dynamic environments (see Power and Schlaggar (2017) for a survey). Recently, several studies (Ba et al., 2016; Zenke et al., 2017b; Miconi et al., 2018, 2019) have been proposed to reformulate the learning rules of neural networks from this perspective. From another perspective of the complementary learning theory (McClelland et al., 1995), it describes that our brain works by a complementary system for fast adaptation through episodic memory (hippocampus) and long-term knowledge consolidation (neocortex). Several attempts (Lüders et al., 2016; Parisi et al., 2018) have been proposed with this theory in mind. However, the proposed learning rules are still not mature, and more seminal methods are expected for long-lasting signs of progress along this journey.

Another exciting direction is to empower a machine learning system or an intelligent agent to be curious to explore unknown and new things by itself. Classical machine learning techniques typically require humans to assign the learning tasks and to provide the corresponding training data. Suppose a machine learning system or a robot interacts with the real-world environment and learn continuously; it needs to identify and formulate its learning tasks and collect its own training data to explore the world. For example, if a robot sees a new person, it should collect some information about the person as positive training data. Another example is that an intelligent chatbot should learn during the conversation, such as expanding its knowledge from previous user utterances, asking the user when it does not understand something, and learning the user's preference. This direction is related to the reinforcement learning paradigm, in which an agent needs to obtain feedback from the environment through trial and error. During the process, the agent often needs to accumulate knowledge from different environments and improve itself to a new environment quickly. Altogether, this direction is very challenging, and studies about it are still limited and immature. We expect more thorough and fundamental research to be conducted to bring the intelligence of an agent or a machine learning system to a new height.

Besides bringing breakthroughs on fundamental theories and methodologies, many other applications are worth studying the lifelong machine learning setup with data efficiency and knowledge retention concerns. Apart from task-oriented dialog systems, recommendation systems, and the image classification task considered in this thesis, it is also valuable to study similar problems for applications such as the fundamental natural language understanding task, open-domain dialog systems, sentiment detection, spam detection, visual tracking and recognition systems, complex control systems, etc. We believe that many practical learning systems will benefit from the rapidly developing lifelong learning techniques in the near future. On the other hand, we envisage that the gradually mature lifelong learning techniques will improve machine learning systems to provide better service and convenience to our society.

Bibliography

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. (2018). Memory aware synapses: Learning what (not) to forget. In *ECCV (3)*, volume 11207 of *Lecture Notes in Computer Science*, pages 144–161. Springer.

Amiri, H. (2019). Neural self-training through spaced repetition. In *NAACL-HLT (1)*, pages 21–31. Association for Computational Linguistics.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In *NIPS*, pages 3981–3989.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *NIPS*, pages 5048–5058.

Arora, G., Rahimi, A., and Baldwin, T. (2019). Does an LSTM forget more than a cnn? an empirical study of catastrophic forgetting in NLP. In *ALTA*, pages 77–86. Australasian Language Technology Association.

Ba, J., Hinton, G. E., Mnih, V., Leibo, J. Z., and Ionescu, C. (2016). Using fast weights to attend to the recent past. In *NIPS*, pages 4331–4339.

Bakhtin, A., Szlam, A., Ranzato, M., and Grave, E. (2018). Lightweight adaptive mixture of neural and n-gram language models. *CoRR*, abs/1804.07705.

Begleiter, R., El-Yaniv, R., and Yona, G. (2004). On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, pages 385–421.

Belouadah, E. and Popescu, A. (2019). IL2M: class incremental learning with dual memory. In *ICCV*, pages 583–592. IEEE.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. (2020). Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *ICLR*.

Berthelot, D., Carlini, N., Goodfellow, I. J., Papernot, N., Oliver, A., and Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. In *NeurIPS*, pages 5050–5060.

Blakeman, S. and Mareschal, D. (2020). A complementary learning systems approach to temporal difference learning. *Neural Networks*, 122:218–230.

Bottou, L. and Cun, Y. (2003). Large scale online learning. *NIPS*, 16:217–224.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *CoRR*, abs/2005.14165.

Budzianowski, P. and Vulic, I. (2019). Hello, it's GPT-2 - how can I help you? towards the use of pretrained language models for task-oriented dialogue systems. In *NGT@EMNLP-IJCNLP*, pages 15–22. Association for Computational Linguistics.

Budzianowski, P., Wen, T., Tseng, B., Casanueva, I., Ultes, S., Ramadan, O., and Gasic, M. (2018). Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*, pages 5016–5026. Association for Computational Linguistics.

Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-end incremental learning. In *ECCV (12)*, volume 11216 of *Lecture Notes in Computer Science*, pages 241–257. Springer.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. (2019). Continual learning with tiny episodic memories. *CoRR*, abs/1902.10486.

Chen, C., Zhang, Y., and Gao, Y. (2018). Learning how to self-learn: Enhancing self-training using neural reinforcement learning. In *IALP*, pages 25–30. IEEE.

Chen, J., Yang, Z., and Yang, D. (2020a). Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*, pages 2147–2157. Association for Computational Linguistics.

Chen, S., Hou, Y., Cui, Y., Che, W., Liu, T., and Yu, X. (2020b). Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *EMNLP (1)*, pages 7870–7881. Association for Computational Linguistics.

Chen, Z. and Liu, B. (2018). Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207.

Chronopoulou, A., Baziotis, C., and Potamianos, A. (2019). An embarrassingly simple approach for transfer learning from pretrained language models. In *NAACL-HLT (1)*, pages 2089–2095. Association for Computational Linguistics.

Cohen, G., Sapiro, G., and Giryes, R. (2018). DNN or k-nn: That is the generalize vs. memorize question. *CoRR*, abs/1805.06822.

Cui, Y., Jia, M., Lin, T., Song, Y., and Belongie, S. J. (2019). Class-balanced loss based on effective number of samples. In *CVPR*, pages 9268–9277. Computer Vision Foundation / IEEE.

Dempster, F. N. (1989). Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019a). BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019b). BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Dimitrakakis, C. (2010). Bayesian variable order markov models. In *AISTATS*, volume 9 of *JMLR Proceedings*, pages 161–168. JMLR.org.

Du, J., Grave, E., Gunel, B., Chaudhary, V., Celebi, O., Auli, M., Stoyanov, V., and Conneau, A. (2020). Self-training improves pre-training for natural language understanding. *CoRR*, abs/2010.02194.

Edunov, S., Ott, M., Auli, M., and Grangier, D. (2018). Understanding back-translation at scale. In *EMNLP*, pages 489–500. Association for Computational Linguistics.

Eric, M., Goel, R., Paul, S., Sethi, A., Agarwal, S., Gao, S., Kumar, A., Goyal, A. K., Ku, P., and Hakkani-Tür, D. (2020). Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *LREC*, pages 422–428. European Language Resources Association.

Fan, A., Lewis, M., and Dauphin, Y. N. (2018). Hierarchical neural story generation. In *ACL (1)*, pages 889–898. Association for Computational Linguistics.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

French, R. M. and Chater, N. (2002). Using noise to compute error surfaces in connectionist networks: A novel means of reducing catastrophic forgetting. *Neural Comput.*, 14(7):1755–1769.

Garcin, F., Dimitrakakis, C., and Faltings, B. (2013). Personalized news recommendation with context trees. In *RecSys*, pages 105–112. ACM.

Garcin, F., Faltings, B., Donatsch, O., Alazzawi, A., Bruttin, C., and Huber, A. (2014). Offline and online evaluation of news recommender systems at swissinfo.ch. In *RecSys*, pages 169–176. ACM.

Garg, D., Gupta, P., Malhotra, P., Vig, L., and Shroff, G. (2019). Sequence and time aware neighborhood for session-based recommendations: Stan. In *SIGIR*, pages 1069–1072. ACM.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *NIPS*, pages 2672–2680.

Grave, E., Cisse, M. M., and Joulin, A. (2017a). Unbounded cache model for online language modeling with open vocabulary. In *NIPS*, pages 6042–6052.

Grave, E., Joulin, A., and Usunier, N. (2017b). Improving neural language models with a continuous cache. In *ICLR (Poster)*. OpenReview.net.

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

Gu, J., Wang, Y., Chen, Y., Li, V. O. K., and Cho, K. (2018). Meta-learning for low-resource neural machine translation. In *EMNLP*, pages 3622–3631. Association for Computational Linguistics.

Ha, D., Dai, A. M., and Le, Q. V. (2017). Hypernetworks. In *ICLR (Poster)*. OpenReview.net.

Ham, D., Lee, J., Jang, Y., and Kim, K. (2020). End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *ACL*, pages 583–592. Association for Computational Linguistics.

Härdle, W. and Linton, O. (1994). Applied nonparametric methods. *Handbook of Econometrics*, 4:2295–2339.

He, H., Chen, S., Li, K., and Xu, X. (2011). Incremental learning from stream data. *IEEE Trans. Neural Networks*, 22(12):1901–1914.

He, J., Gu, J., Shen, J., and Ranzato, M. (2020). Revisiting self-training for neural sequence generation. In *ICLR*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society.

He, R. and McAuley, J. J. (2016). Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*, pages 191–200. IEEE Computer Society.

He, T., Liu, J., Cho, K., Ott, M., Liu, B., Glass, J. R., and Peng, F. (2019a). Mix-review: Alleviate forgetting in the pretrain-finetune framework for neural language generation models. *CoRR*, abs/1910.07117.

He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019b). Bag of tricks for image classification with convolutional neural networks. In *CVPR*, pages 558–567. Computer Vision Foundation / IEEE.

Henderson, M., Thomson, B., and Williams, J. D. (2014). The second dialog state tracking challenge. In *SIGDIAL Conference*, pages 263–272. Association for Computational Linguistics.

Henderson, M., Vulic, I., Gerz, D., Casanueva, I., Budzianowski, P., Coope, S., Spithourakis, G., Wen, T., Mrksic, N., and Su, P. (2019). Training neural response selection for task-oriented dialogue systems. In *ACL (1)*, pages 5392–5404. Association for Computational Linguistics.

Hidasi, B. and Karatzoglou, A. (2018a). Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, pages 843–852. ACM.

Hidasi, B. and Karatzoglou, A. (2018b). Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, pages 843–852. ACM.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In *ICLR (Poster)*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Hinton, G. E., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Hosseini-Asl, E., McCann, B., Wu, C., Yavuz, S., and Socher, R. (2020). A simple language model for task-oriented dialogue. *CoRR*, abs/2005.00796.

Hou, S., Pan, X., Loy, C. C., Wang, Z., and Lin, D. (2019). Learning a unified classifier incrementally via rebalancing. In *CVPR*, pages 831–839. Computer Vision Foundation / IEEE.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. In *ACL (1)*, pages 328–339. Association for Computational Linguistics.

III, H. J. S. (1965). Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inf. Theory*, 11(3):363–371.

Jannach, D. and Ludewig, M. (2017). When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys*, pages 306–310. ACM.

Jégou, H., Douze, M., and Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128.

Kabbur, S., Ning, X., and Karypis, G. (2013). Fism: factored item similarity models for top-n recommender systems. In *KDD*, pages 659–667. ACM.

Kahn, J., Lee, A., and Hannun, A. (2020). Self-training for end-to-end speech recognition. In *ICASSP*, pages 7084–7088. IEEE.

Kaiser, L., Nachum, O., Roy, A., and Bengio, S. (2017). Learning to remember rare events. In *ICLR (Poster)*. OpenReview.net.

Kang, W. and McAuley, J. J. (2018). Self-attentive sequential recommendation. In *ICDM*, pages 197–206. IEEE Computer Society.

Kemker, R., McClure, M., Abitino, A., Hayes, T. L., and Kanan, C. (2018). Measuring catastrophic forgetting in neural networks. In *AAAI*, pages 3390–3398. AAAI Press.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models. In *ICLR*. OpenReview.net.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *ICLR*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Koch, G. (2015). Siamese neural networks for one-shot image recognition.

Kohonen, T. (1984). *Self-organization and associative memory.* Springer-Verlag.

Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456. ACM.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, (8):30–37.

Kozat, S. S., Singer, A. C., and Zeitler, G. (2007). Universal piecewise linear prediction via context trees. *IEEE Trans. Signal Process.*, 55(7-2):3730–3745.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Kumar, A., Bhattamishra, S., Bhandari, M., and Talukdar, P. P. (2019). Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *NAACL-HLT (1)*, pages 3609–3619. Association for Computational Linguistics.

Kumar, M. P., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In *NIPS*, pages 1189–1197. Curran Associates, Inc.

Laine, S. and Aila, T. (2017). Temporal ensembling for semi-supervised learning. In *ICLR (Poster)*.

Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M. A., Tang, L., and Mars, J. (2019). An evaluation dataset for intent classification and out-of-scope prediction. In *EMNLP/IJCNLP (1)*, pages 1311–1316. Association for Computational Linguistics.

Lee, D.-H. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3.

Lee, S. (2017). Toward continual learning for conversational agents. *CoRR*, abs/1712.09943.

Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., and Ma, J. (2017). Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428. ACM.

Li, X., Sun, Q., Liu, Y., Zhou, Q., Zheng, S., Chua, T., and Schiele, B. (2019). Learning to self-train for semi-supervised few-shot classification. In *NeurIPS*, pages 10276–10286.

Li, Y., Zhao, L., Church, K., and Elhoseiny, M. (2020). Compositional language continual learning. In *ICLR*. OpenReview.net.

Li, Z. and Hoiem, D. (2018). Learning without forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(12):2935–2947.

Likhomanenko, T., Xu, Q., Kahn, J., Synnaeve, G., and Collobert, R. (2020). slimipl: Language-model-free iterative pseudo-labeling. *CoRR*, abs/2010.11524.

Liu, Q., Zeng, Y., Mokhosi, R., and Zhang, H. (2018). Stamp: Short-term attention/memory priority model for session-based recommendation. In *KDD*, pages 1831–1839. ACM.

Liu, T., Ungar, L., and Sedoc, J. (2019a). Continual learning for sentence representations using conceptors. In *NAACL-HLT (1)*, pages 3274–3279. Association for Computational Linguistics.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Lomonaco, V. and Maltoni, D. (2017). Core50: a new dataset and benchmark for continuous object recognition. In *CoRL*, volume 78 of *Proceedings of Machine Learning Research*, pages 17–26. PMLR.

Lomonaco, V., Maltoni, D., and Pellegrini, L. (2019). Fine-grained continual learning. *CoRR*, abs/1907.03799.

Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. In *NIPS*, pages 6467–6476.

Lüders, B., Schläger, M., and Risi, S. (2016). Continual learning through evolvable neural turing machines. In *NIPS 2016 Workshop on Continual Learning and Deep Networks (CLDL 2016)*.

Ludewig, M. and Jannach, D. (2018). Evaluation of session-based recommendation algorithms. *User Model. User Adapt. Interact.*, 28(4-5):331–390.

Ma, F., Meng, D., Xie, Q., Li, Z., and Dong, X. (2017). Self-paced co-training. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2275–2284. PMLR.

Maltoni, D. and Lomonaco, V. (2019). Continuous learning in single-incremental-task scenarios. *Neural Networks*, 116:56–73.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. (2017). Pointer sentinel mixture models. In *ICLR (Poster)*. OpenReview.net.

Mi, F., Chen, L., Zhao, M., Huang, M., and Faltings, B. (2020a). Continual learning for natural language generation in task-oriented dialog systems. In *EMNLP (Findings)*, pages 3461–3474. Association for Computational Linguistics.

Mi, F. and Faltings, B. (2017). Adaptive sequential recommendation for discussion forums on moocs using context trees. In *EDM*. International Educational Data Mining Society (IEDMS).

Mi, F. and Faltings, B. (2020). Memory augmented neural model for incremental session-based recommendation. In *IJCAI*, pages 2169–2176. ijcai.org.

Mi, F., Huang, M., Zhang, J., and Faltings, B. (2019). Meta-learning for low-resource natural language generation in task-oriented dialogue systems. In *IJCAI*, pages 3151–3157. ijcai.org.

Mi, F., Kong, L., Lin, T., Yu, K., and Faltings, B. (2020b). Generalized class incremental learning. In *CVPR Workshops*, pages 970–974. IEEE.

Mi, F., Lin, X., and Faltings, B. (2020c). ADER: adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *RecSys*, pages 408–413. ACM.

Miconi, T., Rawal, A., Clune, J., and Stanley, K. O. (2019). Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *ICLR (Poster)*. OpenReview.net.

Miconi, T., Stanley, K. O., and Clune, J. (2018). Differentiable plasticity: training plastic neural networks with backpropagation. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3556–3565. PMLR.

Miller, A. H., Fisch, A., Dodge, J., Karimi, A., Bordes, A., and Weston, J. (2016). Key-value memory networks for directly reading documents. In *EMNLP*, pages 1400–1409. The Association for Computational Linguistics.

Mirzadeh, S., Farajtabar, M., and Ghasemzadeh, H. (2020). Dropout as an implicit gating mechanism for continual learning. In *CVPR Workshops*, pages 945–951. IEEE.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. (2017). Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*.

Miyato, T., Dai, A. M., and Goodfellow, I. J. (2017). Adversarial training methods for semi-supervised text classification. In *ICLR (Poster)*.

Miyato, T., Maeda, S., Koyama, M., and Ishii, S. (2019). Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1979–1993.

Mou, L., Meng, Z., Yan, R., Li, G., Xu, Y., Zhang, L., and Jin, Z. (2016). How transferable are neural networks in NLP applications? In *EMNLP*, pages 479–489. The Association for Computational Linguistics.

Mukherjee, S. and Awadallah, A. H. (2020). Uncertainty-aware self-training for text classification with few labels. *CoRR*, abs/2006.15315.

Munkhdalai, T. and Yu, H. (2017). Meta networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 2554–2563. PMLR.

Naik, D. K. and Mammone, R. (1992). Meta-neural networks that learn by learning. In *IJCNN*, volume 1, pages 437–442. IEEE.

Ng, N., Cho, K., and Ghassemi, M. (2020). SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *EMNLP*, pages 1268–1283. Association for Computational Linguistics.

Nichol, A. and Schulman, J. (2018). Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*.

Niu, Y., Jiao, F., Zhou, M., Yao, T., Xu, J., and Huang, M. (2020). A self-training method for machine reading comprehension with soft evidence extraction. In *ACL*, pages 3916–3927. Association for Computational Linguistics.

O'Reilly, R. C. and Norman, K. A. (2002). Hippocampal and neocortical contributions to memory: Advances in the complementary learning systems framework. *Trends in cognitive sciences*, 6(12):505–510.

Orhan, E. (2018). A simple cache model for image recognition. In *NeurIPS*, pages 10107–10116.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Parisi, G. I., Ji, X., and Wermter, S. (2018). On the role of neurogenesis in overcoming catastrophic forgetting. *CoRR*, abs/1811.02113.

Park, D. S., Zhang, Y., Jia, Y., Han, W., Chiu, C., Li, B., Wu, Y., and Le, Q. V. (2020). Improved noisy student training for automatic speech recognition. In *INTERSPEECH*, pages 2817–2821. ISCA.

Paul, S., Goel, R., and Hakkani-Tür, D. (2019). Towards universal dialogue act tagging for task-oriented dialogues. In *INTERSPEECH*, pages 1453–1457. ISCA.

Peng, B., Li, C., Li, J., Shayandeh, S., Liden, L., and Gao, J. (2020a). SOLOIST: few-shot task-oriented dialog with A single pre-trained auto-regressive model. *CoRR*, abs/2005.05298.

Peng, B., Zhu, C., Li, C., Li, X., Li, J., Zeng, M., and Gao, J. (2020b). Few-shot natural language generation for task-oriented dialog. In *EMNLP (Findings)*, pages 172–182. Association for Computational Linguistics.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Power, J. D. and Schlaggar, B. L. (2017). Neural plasticity across the lifespan. *Wiley Interdisciplinary Reviews: Developmental Biology*, 6(1):e216.

Qian, K. and Yu, Z. (2019). Domain adaptive dialog generation via meta learning. In *ACL (1)*, pages 2639–2649. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Ramalho, T. and Garnelo, M. (2019). Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *ICLR (Poster)*. OpenReview.net.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *ICLR*. OpenReview.net.

Rebuffi, S., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *CVPR*, pages 5533–5542. IEEE Computer Society.

Ren, P., Chen, Z., Li, J., Ren, Z., Ma, J., and de Rijke, M. (2019). Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *AAAI*, pages 4806–4813. AAAI Press.

Ren, Z., Yeh, R., and Schwing, A. (2020). Not all unlabeled data are equal: learning to weight data in semi-supervised learning. In *NeurIPS*, volume 33.

Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM.

Riemer, M., Klinger, T., Bouneffouf, D., and Franceschini, M. (2019). Scalable recollections for continual lifelong learning. In *AAAI*, volume 33, pages 1352–1359. AAAI Press.

Ring, M. B. (1994). *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin Austin, Texas 78712.

Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. (2015). Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *CoRR*, abs/1606.04671.

Saad, D. (1998). Online algorithms and stochastic approximations. *Online Learning*, 5:6–3.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016a). Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning (ICML)*, pages 1842–1850.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016b). One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065*.

Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295. ACM.

Saunders, D. and Byrne, B. (2020). Reducing gender bias in neural machine translation as a domain adaptation problem. In *ACL*, pages 7724–7736. Association for Computational Linguistics.

Saunders, D., Stahlberg, F., de Gispert, A., and Byrne, B. (2019). Domain adaptive inference for neural machine translation. In *ACL*, pages 222–228. Association for Computational Linguistics.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2016). Prioritized experience replay. In *ICLR (Poster)*.

Shah, P., Hakkani-Tür, D., Liu, B., and Tür, G. (2018). Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *NAACL-HLT (3)*, pages 41–51. Association for Computational Linguistics.

Shani, G., Heckerman, D., and Brafman, R. I. (2005). An mdp-based recommender system. *J. Mach. Learn. Res.*, 6:1265–1295.

Shen, Y., Zeng, X., and Jin, H. (2019). A progressive model to enable continual learning for semantic slot filling. In *EMNLP/IJCNLP (1)*, pages 1279–1284. Association for Computational Linguistics.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *NIPS*, pages 2990–2999.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR (Workshop Poster)*.

Smilkov, D., Thorat, N., Kim, B., Viégas, F. B., and Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087.

Sprechmann, P., Jayakumar, S. M., Rae, J. W., Pritzel, A., Badia, A. P., Uria, B., Vinyals, O., Hassabis, D., Pascanu, R., and Blundell, C. (2018). Memory-based parameter adaptation. In *ICLR (Poster)*. OpenReview.net.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Su, J., Maji, S., and Hariharan, B. (2020). When does self-supervision improve few-shot learning? In *ECCV (7)*, volume 12352 of *Lecture Notes in Computer Science*, pages 645–666. Springer.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *NIPS*, pages 2440–2448.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., and Jiang, P. (2019). Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pages 1441–1450. ACM.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H. S., and Hospedales, T. M. (2018). Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208. IEEE Computer Society.

Synnaeve, G., Xu, Q., Kahn, J., Grave, E., Likhomanenko, T., Pratap, V., Sriram, A., Liptchinsky, V., and Collobert, R. (2019). End-to-end ASR: from supervised to semi-supervised learning with modern architectures. *CoRR*, abs/1911.08460.

Tan, Y. K., Xu, X., and Liu, Y. (2016). Improved recurrent neural networks for session-based recommendations. In *DLRS@RecSys*, pages 17–22. ACM.

Thompson, B., Gwinnup, J., Khayrallah, H., Duh, K., and Koehn, P. (2019). Overcoming catastrophic forgetting during domain adaptation of neural machine translation. In *NAACL-HLT (1)*, pages 2062–2068. Association for Computational Linguistics.

Thulasidasan, S., Chennupati, G., Bilmes, J. A., Bhattacharya, T., and Michalak, S. (2019). On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, pages 13888–13899.

Tran, V. and Nguyen, L. (2017). Natural language generation for spoken dialogue system using RNN encoder-decoder networks. In *CoNLL*, pages 442–451. Association for Computational Linguistics.

Tran, V. and Nguyen, L. (2018a). Adversarial domain adaptation for variational neural language generation in dialogue systems. In *COLING*, pages 1205–1217. Association for Computational Linguistics.

Tran, V. and Nguyen, L. (2018b). Dual latent variable model for low-resource natural language generation in dialogue systems. In *CoNLL*, pages 21–30. Association for Computational Linguistics.

Tseng, B., Kreyssig, F., Budzianowski, P., Casanueva, I., Wu, Y., Ultes, S., and Gasic, M. (2018). Variational cross-domain natural language generation for spoken dialogue systems. In *SIGDIAL Conference*, pages 338–343. Association for Computational Linguistics.

Tu, Z., Liu, Y., Shi, S., and Zhang, T. (2018). Learning to remember translation history with a continuous cache. *Trans. Assoc. Comput. Linguistics*, 6:407–420.

Varis, D. and Bojar, O. (2019). Unsupervised pretraining for neural machine translation using elastic weight consolidation. In *ACL (2)*, pages 130–135. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NIPS*, pages 5998–6008.

Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., and Bengio, Y. (2019). Manifold mixup: Better representations by interpolating hidden states. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 6438–6447. PMLR.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. In *NIPS*, pages 3630–3638.

Wei, J. W. and Zou, K. (2019). EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP/IJCNLP (1)*, pages 6381–6387. Association for Computational Linguistics.

Weiss, K. R., Khoshgoftaar, T. M., and Wang, D. (2016). A survey of transfer learning. *J. Big Data*, 3:9.

Welling, M. (2009). Herding dynamical weights to learn. In *ICML*, volume 382 of *ACM International Conference Proceeding Series*, pages 1121–1128. ACM.

Wen, T., Gasic, M., Kim, D., Mrksic, N., Su, P., Vandyke, D., and Young, S. J. (2015a). Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGDIAL Conference*, pages 275–284. The Association for Computer Linguistics.

Wen, T., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., Su, P., Vandyke, D., and Young, S. J. (2016). Multi-domain neural network language generation for spoken dialogue systems. In *HLT-NAACL*, pages 120–129. The Association for Computational Linguistics.

Wen, T., Gasic, M., Mrksic, N., Su, P., Vandyke, D., and Young, S. J. (2015b). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*, pages 1711–1721. The Association for Computational Linguistics.

Wen, T., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L. M., Su, P., Ultes, S., and Young, S. J. (2017a). A network-based end-to-end trainable task-oriented dialogue system. In *EACL (1)*, pages 438–449. Association for Computational Linguistics.

Wen, T.-H., Gašic, M., Mrkšic, N., Rojas-Barahona, L. M., Su, P.-H., Vandyke, D., and Young, S. (2015c). Toward multi-domain language generation using recurrent neural networks. In *NIPS (Workshop)*.

Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J., and Rovatsos, M. (2017b). Fog orchestration for iot services: issues, challenges and directions. *IEEE Internet Computing*, 21(2):16–24.

Willems, F. M. J., Shtarkov, Y. M., and Tjalkens, T. J. (1995). The context-tree weighting method: basic properties. *IEEE Trans. Inf. Theory*, 41(3):653–664.

Wu, C., Madotto, A., Hosseini-Asl, E., Xiong, C., Socher, R., and Fung, P. (2019a). Transferable multi-domain state generator for task-oriented dialogue systems. In *ACL (1)*, pages 808–819. Association for Computational Linguistics.

Wu, C.-S., Hoi, S., Socher, R., and Xiong, C. (2020). TOD-BERT: Pre-trained natural language understanding for task-oriented dialogues. In *EMNLP*, pages 917–929. Association for Computational Linguistics.

Wu, J., Li, L., and Wang, W. Y. (2018). Reinforced co-training. In *NAACL-HLT*, pages 1252–1262. Association for Computational Linguistics.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., and Tan, T. (2019b). Session-based recommendation with graph neural networks. In *AAAI*, pages 346–353. AAAI Press.

Wu, Y., Chen, Y., Wang, L., Ye, Y., Liu, Z., Guo, Y., and Fu, Y. (2019c). Large scale incremental learning. In *CVPR*, pages 374–382. Computer Vision Foundation / IEEE.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M., and Le, Q. V. (2019). Unsupervised data augmentation. *CoRR*, abs/1904.12848.

Xie, Q., Luong, M., Hovy, E. H., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *CVPR*, pages 10684–10695. IEEE.

Xu, Y., Zhong, X., Jimeno-Yepes, A. J., and Lau, J. H. (2020). Forget me not: Reducing catastrophic forgetting for domain adaptation in reading comprehension. In *IJCNN*, pages 1–8. IEEE.

Yalniz, I. Z., Jégou, H., Chen, K., Paluri, M., and Mahajan, D. (2019). Billion-scale semi-supervised learning for image classification. *CoRR*, abs/1905.00546.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, pages 189–196. Morgan Kaufmann Publishers / ACL.

Ye, Z., Geng, Y., Chen, J., Chen, J., Xu, X., Zheng, S., Wang, F., Zhang, J., and Chen, H. (2020). Zero-shot text classification via reinforced self-training. In *ACL*, pages 3014–3024. Association for Computational Linguistics.

Yogatama, D., de Masson d'Autume, C., Connor, J., Kociský, T., Chrzanowski, M., Kong, L., Lazaridou, A., Ling, W., Yu, L., Dyer, C., and Blunsom, P. (2019). Learning and evaluating general linguistic intelligence. *CoRR*, abs/1901.11373.

Yoon, J., Kim, T., Dia, O., Kim, S., Bengio, Y., and Ahn, S. (2018). Bayesian model-agnostic meta-learning. In *NeurIPS*, pages 7342–7352.

Yun, S., Han, D., Chun, S., Oh, S. J., Yoo, Y., and Choe, J. (2019). Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pages 6022–6031. IEEE.

Zenke, F., Poole, B., and Ganguli, S. (2017a). Continual learning through synaptic intelligence. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR.

Zenke, F., Poole, B., and Ganguli, S. (2017b). Continual learning through synaptic intelligence. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 3987–3995. PMLR.

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *ICLR (Poster)*. OpenReview.net.

Zhang, M. and Zhou, Z. (2011). Cotrade: Confident co-training with data editing. *IEEE Trans. Syst. Man Cybern. Part B*, 41(6):1612–1626.

Zhang, Y. and Yang, Q. (2017). A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.

Zhang, Z., He, T., Zhang, H., Zhang, Z., Xie, J., and Li, M. (2019). Bag of freebies for training object detection neural networks. *CoRR*, abs/1902.04103.

Zhao, B., Xiao, X., Gan, G., Zhang, B., and Xia, S. (2020). Maintaining discrimination and fairness in class incremental learning. In *CVPR*, pages 13205–13214. IEEE.

Zheng, Y., Liu, S., and Zhou, Z. (2019). Balancing multi-level interactions for session-based recommendation. *arXiv preprint arXiv:1910.13527*.

Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. (2020). Rethinking pre-training and self-training. In *NeurIPS*, volume 33.

FEI MI

INN 134 Station 14 CH-1015 Lausanne Switzerland
Tel: (+41) 788315623 Email: fei.mi@epfl.ch Homepage: `https://lia.epfl.ch/~mi/`

EDUCATION

Swiss Federal Institute of Technology in Lausanne (EPFL) *2015-now*
Ph.D. in Artificial Intelligence Laboratory
Research Area: Task-oriented Dialog; Recommendation System; Data-efficient Learning; Continual Learning
Supervisor: Prof. Boi Faltings (AAAI Fellow)

Hong Kong University of Science and Technology (HKUST) *2013-2015*
Master of Philosofhy (MPhil) in Computer Science, GPA: 3.8/4.3
Research Area: Machine Learning for MOOCs Data Analysis
Supervisor: Prof. Dit-Yan Yeung (Dean)

Hong Kong University of Science and Technology (HKUST) *Junior, Senior (2011-2013)*
Major degree: Computer Science ; Minor degree: General Business ; GPA: 3.8/4.3

Sun Yat-Sen University (SYSU) *Freshman, Sophomore (2009-2011)*
BEng in Computer Science, Overall Score: 90/100

RESEARCH PUBLICATIONS

[NLP & Task-oriented Dialog]

1. **Fei Mi**, Liangwei Chen, Mengjie Zhao, Minlie Huang, Boi Faltings. "*Continual Learning for Natural Language Generation in Task-oriented Dialog Systems*", **Findings of EMNLP, 2020**
2. Mengjie Zhao, Tao Lin, **Fei Mi**, Martin Jaggi, Hinrich Schütze. "*Masking as an Efficient Alternative to Finetuning for Pretrained Language Models*", **EMNLP, 2020**
3. **Fei Mi**, Minlie Huang, Jiyong Zhang, Boi Faltings. "*Meta-Learning for Low-resource Natural Language Generation in Task-oriented Dialogue Systems*", **IJCAI, 2019**
4. Chaitanya K Joshi, **Fei Mi**, Boi Faltings. "*Personalization in Goal-oriented Dialog*", **NIPS Workshop, 2017.**

In Preparation:
"*Self-Training improves Pre-training for Few-shot Learning in Task-oriented Dialog*", **NAACL 2021 Submission**
"*Multi-intent Natural Language Understanding for Task-oriented Dialog*"
"*Leveraging Pre-trained Models for Continual Learning in Task-oriented Dialog*"

[Recommendation System]

1. **Fei Mi**, Xiaoyu Lin, Boi Faltings. "*ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation*", **RecSys, 2020**, ****Best Short Paper Award****
2. **Fei Mi**, Boi Faltings. "*Memory Augmented Neural Model for Incremental Session-based Recommendation*", **IJCAI, 2020**
3. **Fei Mi**, Boi Faltings. "*Adaptive Sequential Recommendation for Discussion Forums on MOOCs using Context Trees*", **EDM, 2017**
4. **Fei Mi**, Boi Faltings "*Adapting to Drifting Preferences in Recommendation*", **NIPS Workshop, 2016**

[Others]

1. **Fei Mi**, Lingjing Kong, Tao Lin, Kaicheng Yu, Boi Faltings. "*Generalized Class Incremental Learning*", **CVPR Workshop, 2020.**

2. **Fei Mi**, Dit-Yan Yeung. "*Probabilistic Graphical Models for Boosting Cardinal and Ordinal Peer Grading in MOOCs*", **AAAI, 2015**

3. **Fei Mi** and Dit-Yan Yeung. "*Temporal Models for Predicting Student Dropout in Massive Open Online Courses*", **ICDM Workshop, 2015**

In Preparation:
"Fast Learning New Knowledge with Representation Memorization"

AWARDS & HONORS

· Best Short Paper Award, RecSys (2020) *09/2020*

· PHD Fellowship, EPFL *2015-now*

· Postgraduate Scholarship, HKUST *2013- 2015*

· Academic Achievement Award for Graduating Students, HKUST *07/2013*

· Academic Excellence Award, HKUST *05/2013*

· Dean's List, HKUST *2012 - 2013*

PROFESSIONAL EXPERIENCES

· PC Member: IJCAI, AAAI, EMNLP

· Journal Reviewer: TIST, TKDE, AI

· Invited Talk at IJCAI Workshop on Humanizing AI, Macao *08/2019*

· Visiting Scholar at the "Conversational AI Group" of Tsinghua University, Beijing *09/2018-02/2019*
– Develop an NLG model using Meta-learning for few-shot learning.

· Sino-Swiss AI Conference Organization Committee (3 times), Beijing & Lausanne *2017-2019*

· Internship at Aidyia Ltd., Hong Kong *06/2012-09/2012*
– Develop a news sentiment analysis module for a trading platform.